US006185598B1

(12) **United States Patent**
Farber et al.

(10) **Patent No.: US 6,185,598 B1**
(45) **Date of Patent: Feb. 6, 2001**

(54) **OPTIMIZED NETWORK RESOURCE LOCATION**

(75) Inventors: **David A. Farber**, Oak View, CA (US); **Richard E. Greer**, Red Lodge, MT (US); **Andrew D. Swart**, Westlake Village; **James A. Balter**, Santa Barbara, both of CA (US)

(73) Assignee: **Digital Island, Inc.**, San Francisco, CA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/021,506**

(22) Filed: **Feb. 10, 1998**

(51) **Int. Cl.**[7] **G06F 15/16**; G06F 13/00

(52) **U.S. Cl.** 709/200; 709/203; 709/226; 709/233; 709/234; 709/235; 709/236; 709/237; 709/239; 709/240; 709/241

(58) **Field of Search** 709/200, 203, 709/223, 224, 226, 233, 234, 235, 238, 239, 240, 241

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,495,570	1/1985	Kitajima et al.	709/105
4,594,704	6/1986	Ollivier	370/217
4,726,017	2/1988	Krum et al.	370/449
4,839,798	6/1989	Eguchi et al.	709/105
4,920,432	4/1990	Eggers	386/96
4,949,187	8/1990	Cohen	386/69
4,949,248	8/1990	Caro	709/203
5,172,413	12/1992	Bradley	348/7
5,253,341	10/1993	Rozmanith	709/219
5,287,537	2/1994	Newmark et al.	712/29
5,291,554	3/1994	Morales	380/211
5,341,477 *	8/1994	Pitkin et al.	709/226
5,371,532	12/1994	Gelman	348/7
5,410,343	4/1995	Coddington	348/7
5,414,455	5/1995	Hooper	348/7
5,442,389	8/1995	Blahut	348/7
5,442,390	8/1995	Hooper	348/7
5,442,749	8/1995	Northcutt	709/219
5,471,622	11/1995	Eadline	707/3
5,475,615	12/1995	Lin	709/226
5,508,732	4/1996	Bottomley	348/7
5,515,511	5/1996	Nguyen	709/219
5,519,435	5/1996	Anderson	348/8
5,528,281	6/1996	Grady	348/7

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

2 281 793	3/1995	(GB) .
WO 97/11429	3/1997	(WO) .

OTHER PUBLICATIONS

Andresen et al., "SWEB: Towards a Scalable World Wide Web Server on Multicomputers", Proceedings of IPPS, Apr. 15, 1996, pp. 850–856.

Mourad et al., "Scalable Web Server Architectures", Proceedings IEEE Symposium on Computers and Communications, Jul. 1, 1997, pp. 12–16.

Adler, "Distributed Coordination Models for Client/Server Computing", Computer, vol. 28, No. 4, Apr. 1, 1995, pp. 14–22.

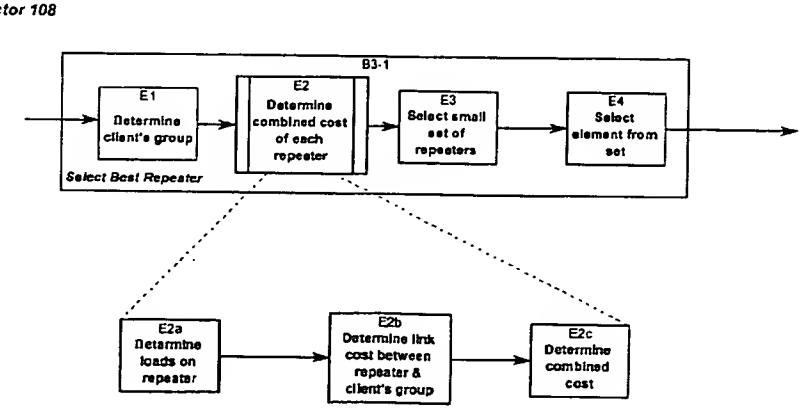
International Search Report dated Jun. 8, 1999 for PCT/US99/01477.

Primary Examiner—Zarni Maung
Assistant Examiner—Almari Romero
(74) *Attorney, Agent, or Firm*—IP Group of Pillsbury Madison & Sutro LLP

(57) **ABSTRACT**

Resource requests made by clients of origin servers in a network are intercepted by reflector mechanisms and selectively reflected to other servers, called repeaters. The reflectors select a best repeater from a set of possible repeaters and redirect the client to the selected best repeater. The client then makes the request of the selected best-repeater. The resource is possibly rewritten to replace at least some of the resource identifiers contained therein with modified resource identifiers designating the repeater instead of the origin server.

27 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

5,539,621	7/1996	Kikinis	361/803
5,544,313	8/1996	Shachnai	709/219
5,544,327	8/1996	Dan	709/234
5,550,577	8/1996	Verbiest	348/7
5,550,863	8/1996	Yurt	375/240
5,550,982	8/1996	Long	709/219
5,557,317	9/1996	Nishio	348/7
5,572,643	11/1996	Judson	709/218
5,590,288	12/1996	Castor	709/201
5,592,611	1/1997	Midgely	714/4
5,594,910	1/1997	Filepp et al.	712/28
5,603,026 *	2/1997	Demers et al.	707/8
5,619,648	4/1997	Canale	709/206
5,623,656	4/1997	Lyons	707/10
5,625,781	4/1997	Cline	345/335
5,627,829	5/1997	Gleeson et al.	370/230
5,630,067	5/1997	Kindell	709/231
5,633,999	5/1997	Clowes	714/6
5,634,006	5/1997	Baugher et al.	709/228
5,644,714	7/1997	Kikinis	709/219
5,649,186	7/1997	Ferguson	707/10
5,659,729	8/1997	Nielsen	707/3
5,666,362	9/1997	Chen	370/420
5,671,279	9/1997	Elgamai	705/79
5,682,512	10/1997	Tetrick	711/202
5,699,513	12/1997	Feigen et al.	713/201
5,715,453	2/1998	Stewart	707/104
5,721,914	2/1998	DeVries	707/104
5,734,831	3/1998	Sanders	709/223
5,742,762	4/1998	Scholl	709/200
5,761,663 *	6/1998	Largarde et al.	709/203
5,774,660 *	6/1998	Brendel et al.	709/239
5,774,668	6/1998	Choquier et al.	709/223
5,796,952	8/1998	Davis	709/224
5,799,141	8/1998	Galipeau et al.	714/13
5,828,847	10/1998	Gehr	709/239
5,870,546 *	2/1999	Kirsch	709/238
5,956,716	9/1999	Kenner	707/10
5,991,809	11/1999	Kriegsman	709/226
6,065,062 *	5/2000	Periasamy et al.	709/242

* cited by examiner

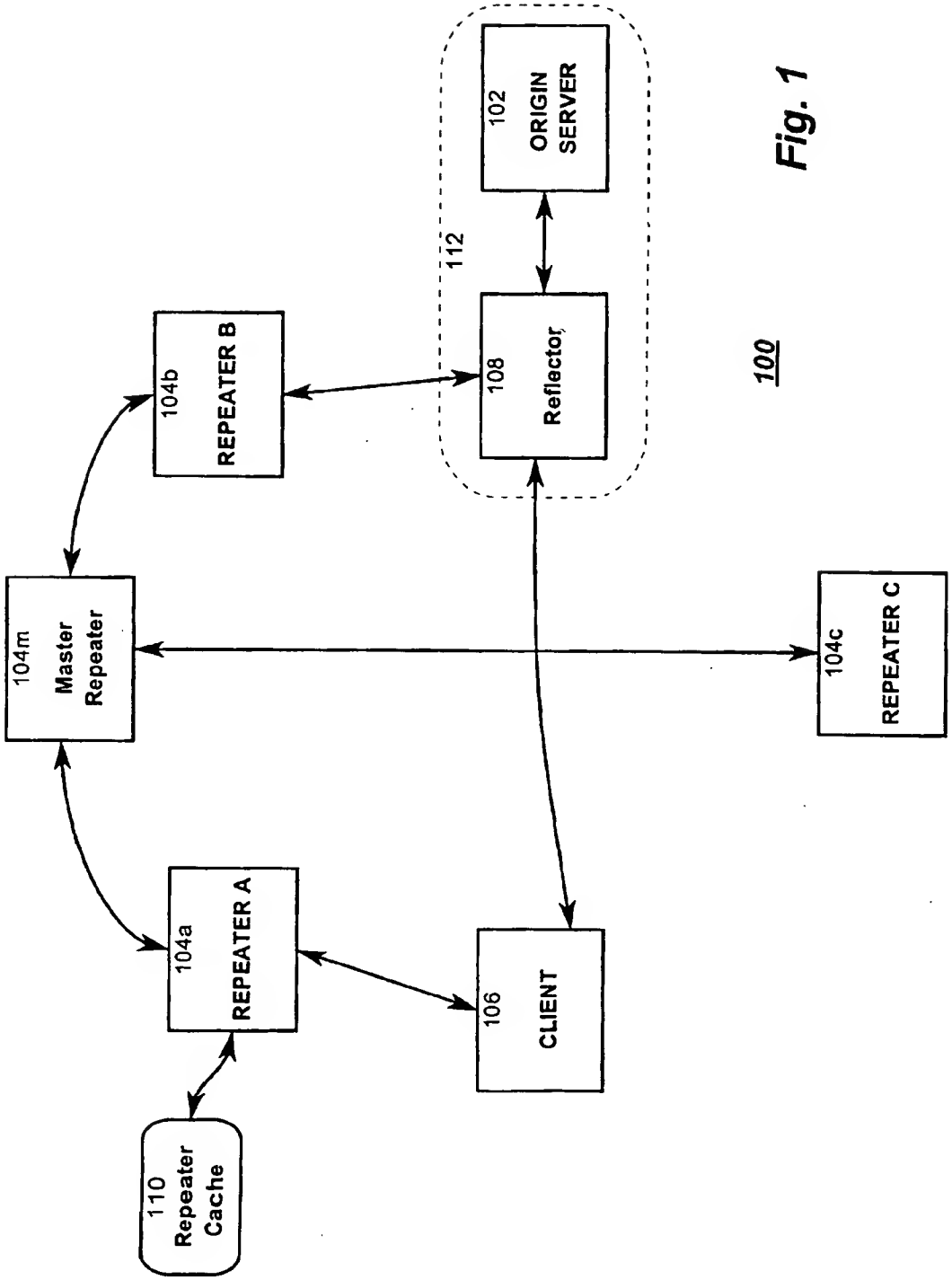

102
ORIGIN SERVER
112
104b
REPEATER B
108
Reflector
104m
Master Repeater
104c
REPEATER C
104a
REPEATER A
106
CLIENT
110
Repeater Cache

Fig. 1

100

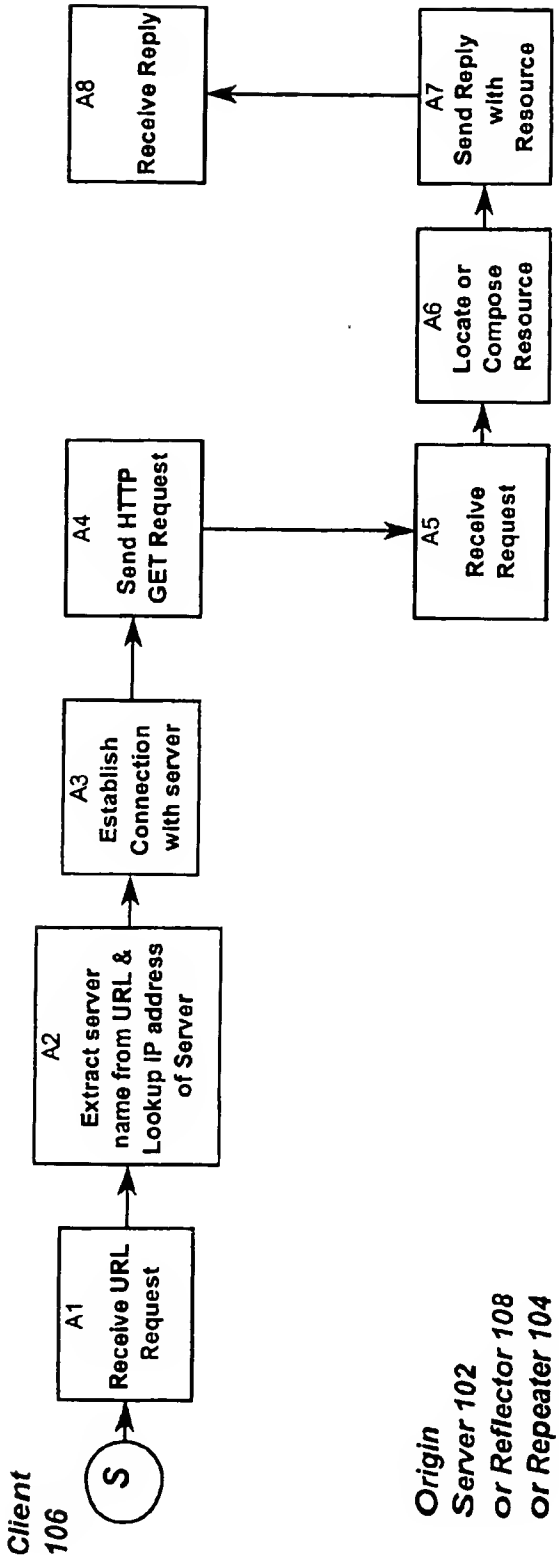

A1
Receive URL Request
A2
Extract server name from URL & Lookup IP address of Server
A3
Establish Connection with server
A4
Send HTTP GET Request
A5
Receive Request
A6
Locate or Compose Resource
A7
Send Reply with Resource
A8
Receive Reply
S
Client 106
Origin Server 102 or Reflector 108 or Repeater 104

Fig. 2

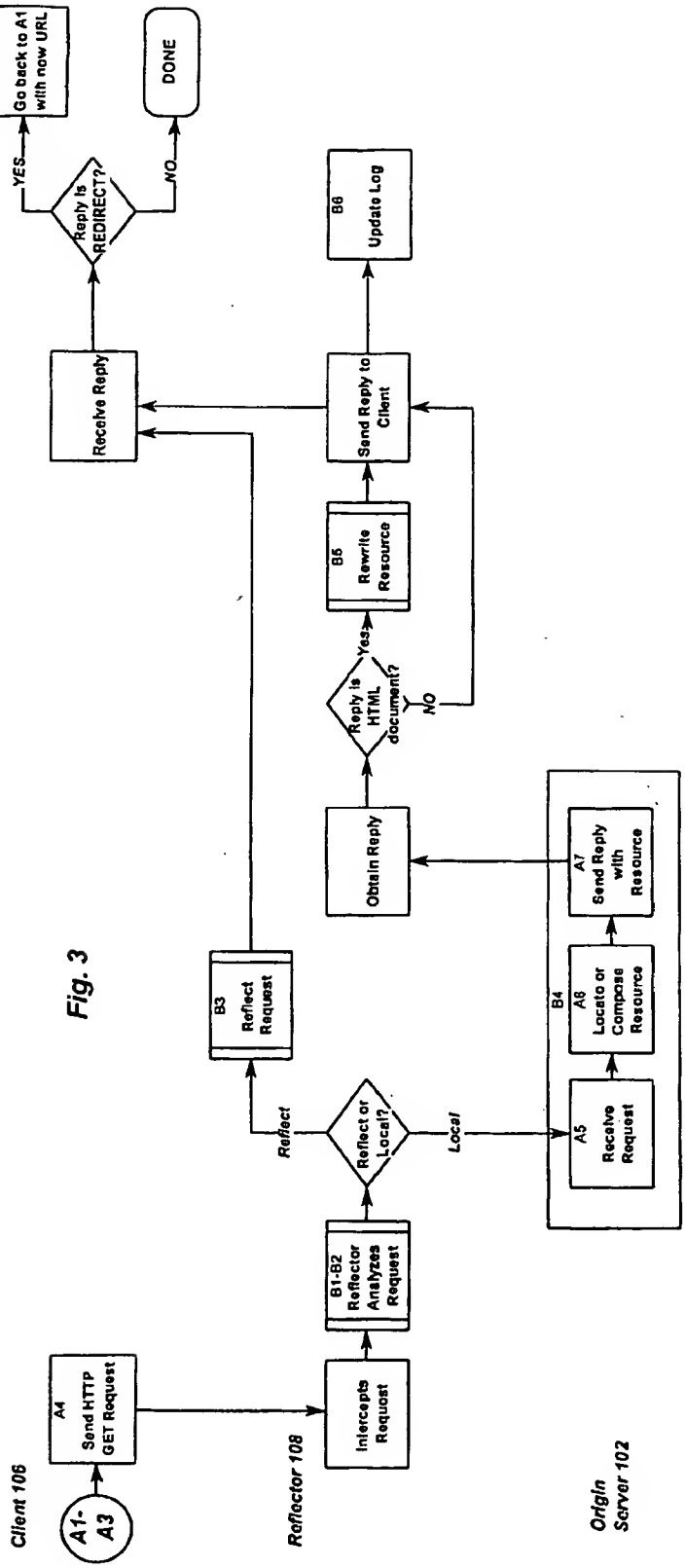

Go back to A1 with new URL
DONE
YES
NO
Reply is REDIRECT?
B6 Update Log
Receive Reply
Send Reply to Client
B5 Rewrite Resource
Yes
NO
Reply is HTML document?
Obtain Reply
A7 Send Reply with Resource
B4
A6 Locate or Compose Resource
A5 Receive Request
B3 Reflect Request
Reflect
Reflect or Local?
Local
B1-B2 Reflector Analyzes Request
A4 Send HTTP GET Request
Intercepts Request
A1-A3
Client 106
Reflector 108
Origin Server 102

Fig. 3

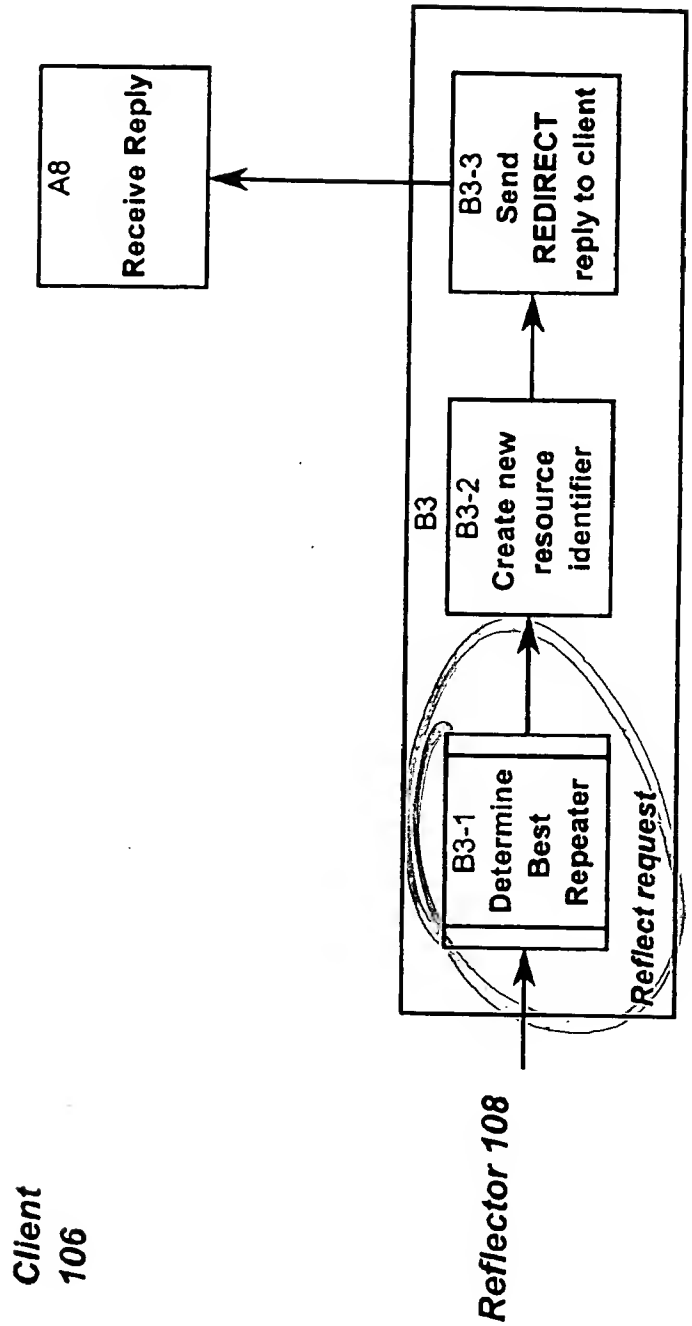

A8
Receive Reply
B3-3
Send
REDIRECT
reply to client
B3
B3-2
Create new
resource
identifier
B3-1
Determine
Best
Repeater
Reflect request
Client
106
Reflector 108

Fig. 4

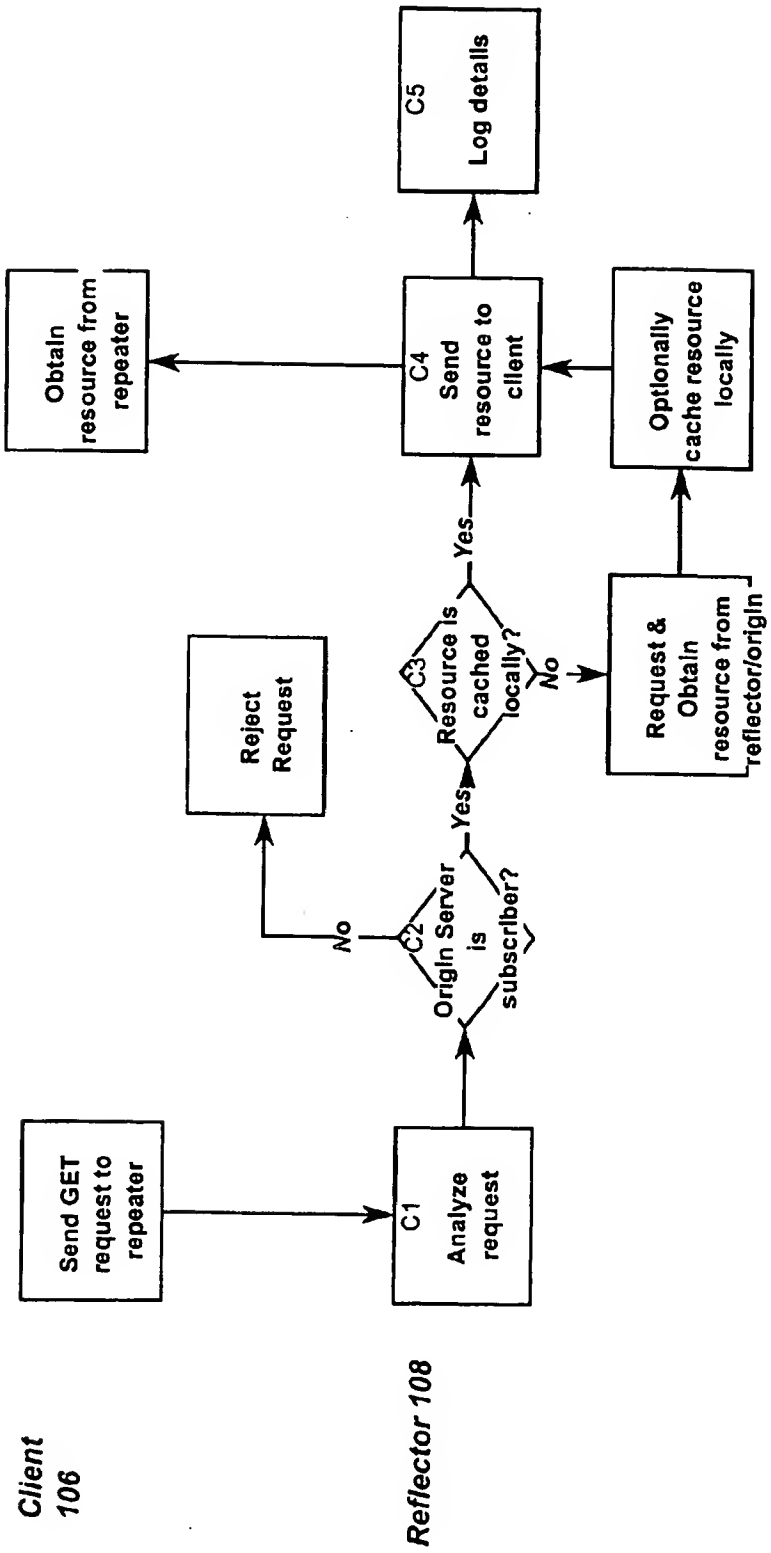

Client 106
Reflector 108
Send GET request to repeater
C1 Analyze request
C2 Origin Server is subscriber?
No
Reject Request
Yes
C3 Resource is cached locally?
No
Request & Obtain resource from reflector/origin
Yes
Optionally cache resource locally
C4 Send resource to client
Obtain resource from repeater
C5 Log details

Fig. 5

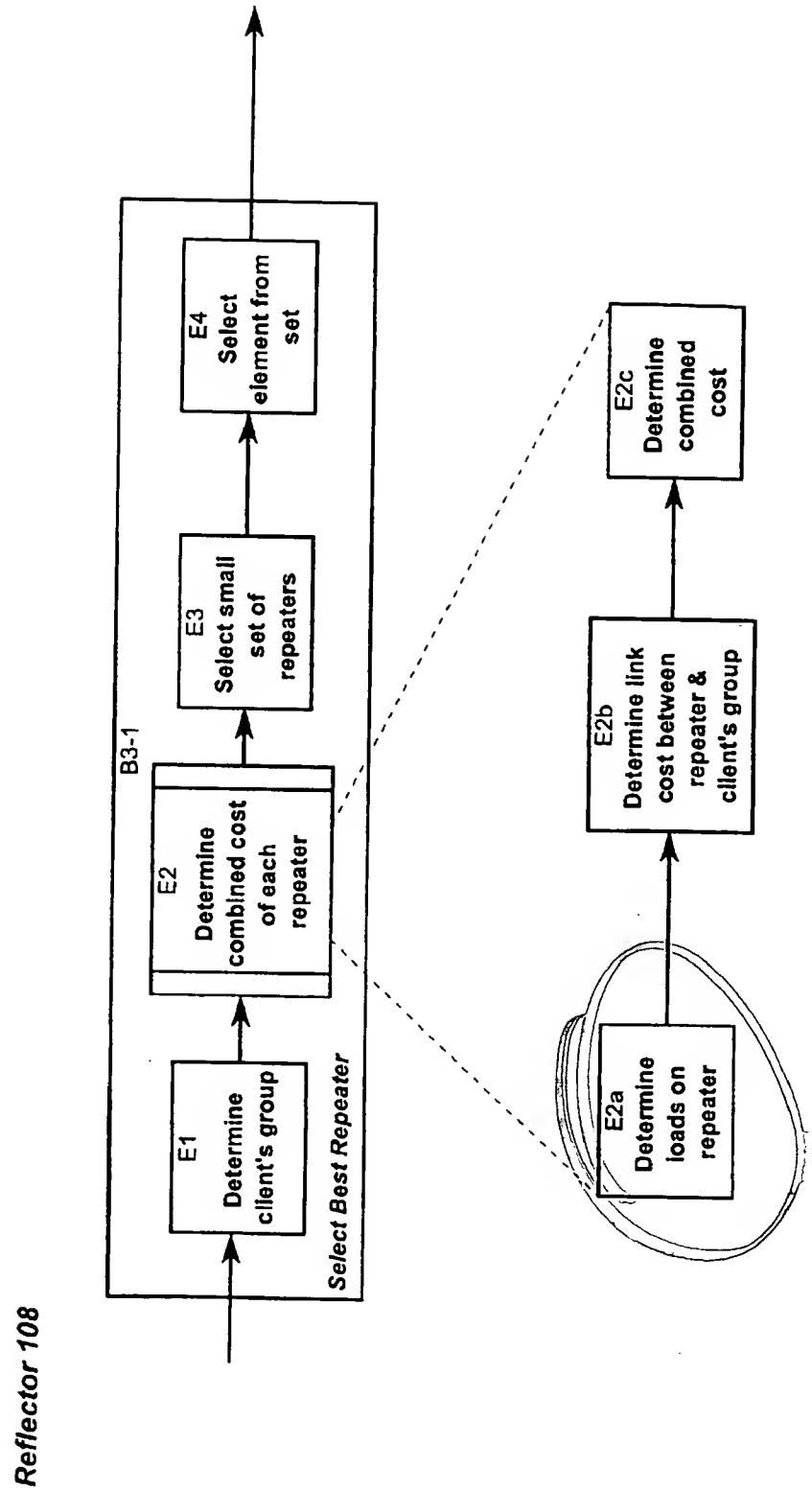

Reflector 108
Select Best Repeater
B3-1
E1 Determine client's group
E2 Determine combined cost of each repeater
E3 Select small set of repeaters
E4 Select element from set
E2a Determine loads on repeater
E2b Determine link cost between repeater & client's group
E2c Determine combined cost

Fig. 6

OPTIMIZED NETWORK RESOURCE LOCATION

FIELD OF THE INVENTION

This invention relates to replication of resources in computer networks.

BACKGROUND OF THE INVENTION

The advent of global computer networks, such as the Internet, have led to entirely new and different ways to obtain information. A user of the Internet can now access information from anywhere in the world, with no regard for the actual location of either the user or the information. A user can obtain information simply by knowing a network address for the information and providing that address to an appropriate application program such as a network browser.

The rapid growth in popularity of the Internet has imposed a heavy traffic burden on the entire network. Solutions to problems of demand (e.g., better accessibility and faster communication links) only increase the strain on the supply. Internet Web sites (referred to here as "publishers") must handle ever-increasing bandwidth needs, accommodate dynamic changes in load, and improve performance for distant browsing clients, especially those overseas. The adoption of content-rich applications, such as live audio and video, has further exacerbated the problem.

To address basic bandwidth growth needs, a Web publisher typically subscribes to additional bandwidth from an Internet service provider (ISP), whether in the form of larger or additional "pipes" or channels from the ISP to the publisher's premises, or in the form of large bandwidth commitments in an ISP's remote hosting server collection. These increments are not always as fine-grained as the publisher needs, and quite often lead times can cause the publisher's Web site capacity to lag behind demand.

To address more serious bandwidth growth problems, publishers may develop more complex and costly custom solutions. The solution to the most common need, increasing capacity, is generally based on replication of hardware resources and site content (known as mirroring), and duplication of bandwidth resources. These solutions, however, are difficult and expensive to deploy and operate. As a result, only the largest publishers can afford them, since only those publishers can amortize the costs over many customers (and Web site hits).

A number of solutions have been developed to advance replication and mirroring. In general, these technologies are designed for use by a single Web site and do not include features that allow their components to be shared by many Web sites simultaneously.

Some solution mechanisms offer replication software that helps keep mirrored servers up-to-date. These mechanisms generally operate by making a complete copy of a file system. One such system operates by transparently keeping multiple copies of a file system in synch. Another system provides mechanisms for explicitly and regularly copying files that have changed. Database systems are particularly difficult to replicate, as they are continually changing. Several mechanisms allow for replication of databases, although there are no standard approaches for accomplishing it. Several companies offering proxy caches describe them as replication tools. However, proxy caches differ because they are operated on behalf of clients rather than publishers.

Once a Web site is served by multiple servers, a challenge is to ensure that the load is appropriately distributed or

balanced among those servers. Domain name-server-based round-robin address resolution causes different clients to be directed to different mirrors.

Another solution, load balancing, takes into account the load at each server (measured in a variety of ways) to select which server should handle a particular request.

Load balancers use a variety of techniques to route the request to the appropriate server. Most of those load-balancing techniques require that each server be an exact replica of the primary Web site. Load balancers do not take into account the "network distance" between the client and candidate mirror servers.

Assuming that client protocols cannot easily change, there are two major problems in the deployment of replicated resources. The first is how to select which copy of the resource to use. That is, when a request for a resource is made to a single server, how should the choice of a replica of the server (or of that data) be made. We call this problem the "rendezvous problem". There are a number of ways to get clients to rendezvous at distant mirror servers. These technologies, like load balancers, must route a request to an appropriate server, but unlike load balancers, they take network performance and topology into account in making the determination.

A number of companies offer products which improve network performance by prioritizing and filtering network traffic. Proxy caches provide a way for client aggregators to reduce network resource consumption by storing copies of popular resources close to the end users. A client aggregator is an Internet service provider or other organization that brings a large number of clients operating browsers to the Internet. Client aggregators may use proxy caches to reduce the bandwidth required to serve web content to these browsers. However, traditional proxy caches are operated on behalf of Web clients rather than Web publishers.

Proxy caches store the most popular resources from all publishers, which means they must be very large to achieve reasonable cache efficiency. (The efficiency of a cache is defined as the number of requests for resources which are already cached divided by the total number of requests.)

Proxy caches depend on cache control hints delivered with resources to determine when the resources should be replaced. These hints are predictive, and are necessarily often incorrect, so proxy caches frequently serve stale data. In many cases, proxy cache operators instruct their proxy to ignore hints in order to make the cache more efficient, even though this causes it to more frequently serve stale data.

Proxy caches hide the activity of clients from publishers. Once a resource is cached, the publisher has no way of knowing how often it was accessed from the cache.

SUMMARY OF THE INVENTION

This invention provides a way for servers in a computer network to off-load their processing of requests for selected resources by determining a different server (a "repeater") to process those requests. The selection of the repeater can be made dynamically, based on information about possible repeaters.

If a requested resource contains references to other resources, some or all of these references can be replaced by references to repeaters.

Accordingly, in one aspect, this invention is a method of processing resource requests in a computer network. First a client makes a request for a particular resource from an origin server, the request including a resource identifier for

the particular resource, the resource identifier sometimes including an indication of the origin server. Requests arriving at the origin server do not always include an indication of the origin server; since they are sent to the origin server, they do not need to name it. A mechanism referred to as a reflector, co-located with the origin server, intercepts the request from the client to the origin server and decides whether to reflect the request or to handle it locally. If the reflector decides to handle the request locally, it forwards it to the origin server, otherwise it selects a "best" repeater to process the request. If the request is reflected, the client is provided with a modified resource identifier designating the repeater.

The client gets the modified resource identifier from the reflector and makes a request for the particular resource from the repeater designated in the modified resource identifier.

When the repeater gets the client's request, it responds by returning the requested resource to the client. If the repeater has a local copy of the resource then it returns that copy, otherwise it forwards the request to the origin server to get the resource, and saves a local copy of the resource in order to serve subsequent requests.

The selection by the reflector of an appropriate repeater to handle the request can be done in a number of ways. In the preferred embodiment, it is done by first pre-partitioning the network into "cost groups" and then determining which cost group the client is in. Next, from a plurality of repeaters in the network, a set of repeaters is selected, the members of the set having a low cost relative to the cost group which the client is in. In order to determine the lowest cost, a table is maintained and regularly updated to define the cost between each group and each repeater. Then one member of the set is selected, preferably randomly, as the best repeater.

If the particular requested resource itself can contain identifiers of other resources, then the resource may be rewritten (before being provided to the client). In particular, the resource is rewritten to replace at least some of the resource identifiers contained therein with modified resource identifiers designating a repeater instead of the origin server. As a consequence of this rewriting process, when the client requests other resources based on identifiers in the particular requested resource, the client will make those requests directly to the selected repeater, bypassing the reflector and origin server entirely.

Resource rewriting must be performed by reflectors. It may also be performed by repeaters, in the situation where repeaters "peer" with one another and make copies of resources which include rewritten resource identifiers that designate a repeater.

In a preferred embodiment, the network is the Internet and the resource identifier is a uniform resource locator (URL) for designating resources on the Internet, and the modified resource identifier is a URL designating the repeater and indicating the origin server (as described in step B3 below), and the modified resource identifier is provided to the client using a REDIRECT message. Note, only when the reflector is "reflecting" a request is the modified resource identifier provided using a REDIRECT message.

In another aspect, this invention is a computer network comprising a plurality of origin servers, at least some of the origin servers having reflectors associated therewith, and a plurality of repeaters.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects and advantages of the invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which the reference characters refer to like parts throughout and in which:

FIG. 1 depicts a portion of a network environment according to the present invention; and

FIGS. 2–6 are flow charts of the operation of the present invention.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EXEMPLARY EMBODIMENTS

Overview

FIG. 1 shows a portion of a network environment **100** according to the present invention, wherein a mechanism (reflector **108**, described in detail below) at a server (herein origin server **102**) maintains and keeps track of a number of partially replicated servers or repeaters **104***a*, **104***b*, and **104***c*. Each repeater **104***a*, **104***b*, and **104***c* replicates some or all of the information available on the origin server **102** as well as information available on other origin servers in the network **100**. Reflector **108** is connected to a particular repeater known as its "contact" repeater ("Repeater B" **104***b* in the system depicted in FIG. 1). Preferably each reflector maintains a connection with a single repeater known as its contact, and each repeater maintains a connection with a special repeater known as its master repeater (e.g., repeater **104***m* for repeaters **104***a*, **104***b* and **104***c* in FIG. 1).

Thus, a repeater can be considered as a dedicated proxy server that maintains a partial or sparse mirror of the origin server **102**, by implementing a distributed coherent cache of the origin server. A repeater may maintain a (partial) mirror of more than one origin server. In some embodiments, the network **100** is the Internet and repeaters mirror selected resources provided by origin servers in response to clients' HTTP hypertext transfer protocol) and FTP (file transfer protocol) requests.

A client **106** connects, via the network **100**, to origin server **102** and possibly to one or more repeaters **104***a* etc.

Origin server **102** is a server at which resources originate. More generally, the origin server **102** is any process or collection of processes that provide resources in response to requests from a client **106**. Origin server **102** can be any off-the-shelf Web server. In a preferred embodiment, origin server **102** is typically a Web server such as the Apache server or Netscape Communications Corporation's Enterprise™ server.

Client **106** is a processor requesting resources from origin server **102** on behalf of an end user. The client **106** is typically a user agent (e.g., a Web browser such as Netscape Communications Corporation's Navigator™) or a proxy for a user agent. Components other than the reflector **108** and the repeaters **104***a*, **104***b*, etc., may be implemented using commonly available software programs. In particular, this invention works with any HTTP client (e.g., a Web browser), proxy cache, and Web server. In addition, the reflector **108** might be fully integrated into the data server **112** (for instance, in a Web Server). These components might be loosely integrated based on the use of extension mechanisms (such as so-called add-in modules) or tightly integrated by modifying the service component specifically to support the repeaters.

Resources originating at the origin server **102** may be static or dynamic. That is, the resources may be fixed or they may be created by the origin server **102** specifically in response to a request. Note that the terms "static" and

"dynamic" are relative, since a static resource may change at some regular, albeit long, interval.

Resource requests from the client **106** to the origin server **102** are intercepted by reflector **108** which for a given request either forwards the request on to the origin server **102** or conditionally reflects it to some repeater **104***a*, **104***b*, etc. in the network **100**. That is, depending on the nature of the request by the client **106** to the origin server **102**, the reflector **108** either serves the request locally (at the origin server **102**), or selects one of the repeaters (preferably the best repeater for the job) and reflects the request to the selected repeater. In other words, the reflector **108** causes requests for resources from origin server **102**, made by client **106**, to be either served locally by the origin server **102** or transparently reflected to the best repeater **104***a*, **104***b*, etc. The notion of a best repeater and the manner in which the best repeater is selected are described in detail below.

Repeaters **104***a*, **104***b*, etc. are intermediate processors used to service client requests thereby improving performance and reducing costs in the manner described herein. Within repeaters **104***a*, **104***b*, etc., are any processes or collections of processes that deliver resources to the client **106** on behalf of the origin server **102**. A repeater may include a repeater cache **110**, used to avoid unnecessary transactions with the origin server **102**.

The reflector **108** is a mechanism, preferably a software program, that intercepts requests that would normally be sent directly to the origin server **102**. While shown in the drawings as separate components, the reflector **108** and the origin server **102** are typically co-located, e.g., on a particular system such as data server **112**. (As discussed below, the reflector **108** may even be a "plug in" module that becomes part of the origin server **102**.

FIG. **1** shows only a part of a network **100** according to this invention. A complete operating network consists of any number of clients, repeaters, reflectors, and origin servers. Reflectors communicate with the repeater network, and repeaters in the network communicate with one another.

Uniform Resource Locators

Each location in a computer network has an address which can generally be specified as a series of names or numbers. In order to access information, an address for that information must be known. For example, on the World Wide Web ("the Web") which is a subset of the Internet, the manner in which information address locations are provided has been standardized into Uniform Resource Locators (URLs). URLs specify the location of resources (information, data files, etc.) on the network.

The notion of URLs becomes even more useful when hypertext documents are used. A hypertext document is one which includes, within the document itself, links (pointers or references) to the document itself or to other documents. For example, in an on-line legal research system, each case may be presented as a hypertext document. When other cases are cited, links to those cases can be provided. In this way, when a person is reading a case, they can follow cite links to read the appropriate parts of cited cases.

In the case of the Internet in general and the World Wide Web specifically, documents can be created using a standardized form known as the Hypertext Markup Language (HTML). In HTML, a document consists of data (text, images, sounds, and the like), including links to other sections of the same document or to other documents. The links are generally provided as URLs, and can be in relative or absolute form. Relative URLs simply omit the parts of the URL which are the same as for the document including the link, such as the address of the document (when linking to the same document), etc. In general, a browser program will fill in missing parts of a URL using the corresponding parts from the current document, thereby forming a fully formed URL including a fully qualified domain name, etc.

A hypertext document may contain any number of links to other documents, and each of those other documents may be on a different server in a different part of the world. For example, a document may contain links to documents in Russia, Africa, China and Australia. A user viewing that document at a particular client can follow any of the links transparently (i.e., without knowing where the document being linked to actually resides). Accordingly, the cost (in terms of time or money or resource allocation) of following one link versus another may be quite significant.

URLs generally have the following form (defined in detail in T. Berners-Lee et al, *Uniform Resource Locators (URL)*, Network Working Group, Request for Comments: 1738, Category: Standards Track, December 1994, located at "http://ds.internic.net/rfc/rfc1738.txt", which is hereby incorporated herein by reference):

```
scheme://host[:port]/url-path
```

where "scheme" can be a symbol such as "file" (for a file on the local system), "ftp" (for a file on an anonymous FTP file server), "http" (for a file on a file on a Web server), and "telnet" (for a connection to a Telnet-based service). Other schemes, can also be used and new schemes are added every now and then. The port number is optional, the system substituting a default port number (depending on the scheme) if none is provided. The "host" field maps to a particular network address for a particular computer. The "url-path" is relative to the computer specified in the "host" field. A url-path is typically, but not necessarily, the pathname of a file in a web server directory.

For example, the following is a URL identifying a file "F" in the path "A/B/C" on a computer at "www.uspto.gov":

```
http://www.uspto.gov/A/B/C/F
```

In order to access the file "F" (the resource) specified by the above URL, a program (e.g., a browser) running on a user's computer (i.e., a client computer) would have to first locate the computer (i.e., a server computer) specified by the host name. I.e., the program would have to locate the server "www.uspto.gov". To do this, it would access a Domain Name Server (DNS), providing the DNS with the host name ("www.uspto.gov"). The DNS acts as a kind of centralized directory for resolving addresses from names. If the DNS determines that there is a (remote server) computer corresponding to the name "www.upsto.gov", it will provide the program with an actual computer network address for that server computer. On the Internet this is called an Internet Protocol (or IP) address and it has the form "123.345.456.678". The program on the user's (client) computer would then use the actual address to access the remote (server) computer.

The program opens a connection to the HTTP server (Web server) on the remote computer "www.uspto.gov" and uses the connection to send a request message to the remote computer (using the HTTP scheme). The message is typically an HTTP GET request which includes the url-path of the requested resource, "A/B/C/F". The HTTP server receives the request and uses it to access the resource specified by the url-path "A/B/C/F". The server returns the resource over the same connectinn.

Thus, conventionally HTTP client requests for Web resources at an origin server **102** are processed as follows

(see FIG. 2) (This is a description of the process when no reflector **108** is installed.):

A1. A browser (e.g., Netscape's Navigator) at the client receives a resource identifier (i.e., a URL) from a user.

A2. The browser extracts the host (origin server) name from the resource identifier, and uses a domain name server (DNS) to look up the network (IP) address of the corresponding server. The browser also extracts a port number, if one is present, or uses a default port number (the default port number for http requests is **80**).

A3. The browser uses the server's network address and port number to establish a connection between the client **106** and the host or origin server **102**.

A4. The client **106** then sends a (GET) request over the connection identifying the requested resource.

A5. The origin server **102** receives the request and

A6. locates or composes the corresponding resource.

A7. The origin server **102** then sends back to the client **106** a reply containing the requested resource (or some form of error indicator if the resource is unavailable). The reply is sent to the client over the same connection as that on which the request was received from the client.

A8. The client **106** receives the reply from the origin server **102**.

There are many variations of this basic model. For example, in one variation, instead of providing the client with the resource, the origin server can tell the client to re-request the resource by another name. To do so, in A7 the server **102** sends back to the client **106** a reply called a "REDIRECT" which contains a new URL indicating the other name. The client **106** then repeats the entire sequence, normally without any user intervention, this time requesting the resource identified by the new URL.

System Operation

In this invention reflector **108** effectively takes the place of an ordinary Web server or origin server **102**. The reflector **108** does this by taking over the origin server's IP address and port number. In this way, when a client tries to connect to the origin server **102**, it will actually connect to the reflector **108**. The original Web server (or origin server **102**) must then accept requests at a different network (IP) address, or at the same IP address but on a different port number. Thus, using this invention, the server referred to in A3–A7 above is actually a reflector **108**.

Note that it is also possible to leave the origin server's network address as it is and to let the reflector run at a different address or on a different port. In this way the reflector does not intercept requests sent to the origin server, but can still be sent requests addressed specifically to the reflector. Thus the system can be tested and configured without interrupting its normal operation. The reflector **108** supports the processing as follows (see FIG. 3): upon receipt of a request,

B1. The reflector **108** analyzes the request to determine whether or not to reflect the request. To do this, first the reflector determines whether the sender (client **106**) is a browser or a repeater. Requests issued by repeaters must be served locally by the origin server **102**. This determination can be made by looking up the network (IP) address of the sender in a list of known repeater network (IP) addresses. Alternatively, this determination could be made by attaching information to a request to indicate that the request is from a specific repeater, or repeaters can request resources from a special port other than the one used for ordinary clients.

B2. If the request is not from a repeater, the reflector looks up the requested resource in a table (called the "rule base") to determine whether the resource requested is "repeatable". Based on this determination, the reflector either reflects the request (B3, described below) or serves the request locally (B4, described below).

The rule base is a list of regular expressions and associated attributes. (Regular expressions are well-known in the field of computer science. A small bibliography of their use is found in Abo, et al., "Compilers, Principles, techniques and tools", Addison-Wesley, 1986, pp. 157–158.) The resource identifier (URL) for a given request is looked up in the rule base by matching it sequentially with each regular expression. The first match identifies the attributes for the resource, namely repeatable or local. If there is no match in the rule base, a default attribute is used. Each reflector has its own rule base, which is manually configured by the reflector operator.

B3. To reflect a request, (to serve a request locally go to B4), as shown in FIG. 4, the reflector determines (B3-1) the best repeater to reflect the request to, as described in detail below. The reflector then creates (B3-2) a new resource identifier (URL) (using the requested URL and the best repeater) that identifies the same resource at the selected repeater.

It is necessary that the reflection step create a single URL containing the URL of the original resource, as well as the identity of the selected repeater. A special form of URL is created to provide this information. This is done by creating a new URL as follows:

D1. Given a repeater name, scheme, origin server name and path, create a new URL. If the scheme is "http", the preferred embodiment uses the following format:

```
http://<repeater>/<server>/<path>
```

If the form used is other than "http", the preferred embodiment uses the following format:

```
http://<repeater>/<server>@proxy=<scheme>@/<path>
```

The reflector can also attach a MIME type to the request, to cause the repeater to provide that MIME type with the result. This is useful because many protocols (such as FTP) do not provide a way to attach a MIME type to a resource. The format is

```
http://<repeater>/<server>@proxy=<scheme>:<type>@/<path>
```

This URL is interpreted when received by the repeater.

The reflector then sends (B3-3) a REDIRECT reply containing this new URL to the requesting client. The HTTP REDIRECT command allows the reflector to send the browser a single URL to retry the request.

B4. To serve a request locally, the request is sent by the reflector to ("forwarded to") the origin server **102**. In this mode, the reflector acts as a reverse proxy server. The origin server **102** processes the request in the normal manner (A5–A7). The reflector then obtains the origin server's reply to the request which it inspects to determine if the requested resource is an HTML document, i.e., whether the requested resource is one which itself contains resource identifiers.

B5. If the resource is an HTML document then the reflector rewrites the HTML document by modifying resource identifiers (URLs) within it, as described below. The resource, possibly as modified by rewriting, is then returned in a reply to the requesting client **106**.

If the requesting client is a repeater, the reflector may temporarily disable any cache-control modifiers which the origin server attached to the reply. These disabled cache-control modifiers are later re-enabled when the content is served from the repeater. This mechanism makes it possible for the origin server to prevent resources from being cached at normal proxy caches, without affecting the behavior of the cache at the repeater.

B6. Whether the request is reflected or handled locally, details about the transaction, such as the current time, the address of the requester, the URL requested, and the type of response generated, are written by the reflector to a local log file.

By using a rule base (B2), it is possible to selectively reflect resources. There are a number of reasons that certain particular resources cannot be effectively repeated (and therefore should not be reflected), for instance:

the resource is composed uniquely for each request;

the resource relies on a so-called cookie (browsers will not send cookies to repeaters with different domain names);

the resource is actually a program (such as a Java applet) that will run on the client and that wishes to connect to a service (Java requires that the service be running on the same machine that provided the applet).

In addition, the reflector **108** can be configured so that requests from certain network addresses (e.g., requests from clients on the same local area network as the reflector itself) are never reflected. Also, the reflector may choose not to reflect requests because the reflector is exceeding its committed aggregate information rate, as described below.

A request which is reflected is automatically mirrored at the repeater when the repeater receives and processes the request.

The combination of the reflection process described here and the caching process described below effectively creates a system in which repeatable resources are migrated to and mirrored at the selected reflector, while non-repeatable resources are not mirrored.

Alternate Approach

Placing the origin server name in the reflected URL is generally a good strategy, but it may be considered undesirable for aesthetic or (in the case, e.g., of cookies) certain technical reasons.

It is possible to avoid the need for placing both the repeater name and the server name in the URL. Instead, a "family" of names may be created for a given origin server, each name identifying one of the repeaters used by that server.

For instance, if www.example.com is the origin server, names for three repeaters might be created:

wr1.example.com

wr2.example.com

wr3.example.com

The name "wr1.example.com" would be an alias for repeater 1, which might also be known by other names such as "wr1.anotherExample.com" and "wr1.example.edu".

If the repeater can determine by which name it was addressed, it can use this information (along with a table that associates repeater alias names with origin server names) to determine which origin server is being addressed. For instance, if repeater 1 is addressed as wr1.example.com, then the origin server is "www.example.com"; if it is addressed as "wr1.anotherExample.com", then the origin server is "www.anotherExample.com".

The repeater can use two mechanisms to determine by which alias it is addressed:

1. Each alias can be associated with a different IP address. Unfortunately, this solution does not scale well, as IP addresses are currently scarce, and the number of IP addresses required grows as the product of origin servers and repeaters.

2. The repeater can attempt to determine the alias name used by inspecting the "host:" tag in the HTTP header of the request. Unfortunately, some old browsers still in use do not attach the "host:" tag to a request. Reflectors would need to identify such browsers (the browser identity is a part of each request) and avoid this form of reflection.

How a Repeater Handles a Request

When a browser receives a REDIRECT response (as produced in B3), it reissues a request for the resource using the new resource identifier (URL) (A1–A5). Because the new identifier refers to a repeater instead of the origin server, the browser now sends a request for the resource to the repeater which processes a request as follows, with reference to FIG. 5:

C1. First the repeater analyzes the request to determine the network address of the requesting client and the path of the resource requested. Included in the path is an origin server name (as described above with reference to B3).

C2. The repeater uses an internal table to verify that the origin server belongs to a known "subscriber". A subscriber is an entity (e.g., a company) that publishes resources (e.g., files) via one or more origin servers. When the entity subscribes, it is permitted to utilize the repeater network. The subscriber tables described below include the information that is used to link reflectors to subscribers.

If the request is not for a resource from a known subscriber, the request is rejected. To reject a request, the repeater returns a reply indicating that the requested resource does not exist.

C3. The repeater then determines whether the requested resource is cached locally. If the requested resource is in the repeater's cache it is retrieved. On the other hand, if a valid copy of the requested resource is not in the repeater's cache, the repeater modifies the incoming URL, creating a request that it issues directly to the originating reflector which processes it (as in B1–B6). Because this request to the originating reflector is from a repeater, the reflector always returns the requested resource rather than reflecting the request. (Recall that reflectors always handle requests from repeaters locally.) If the repeater obtained the resource from the origin server, the repeater then caches the resource locally.

If a resource is not cached locally, the cache can query its "peer caches" to see if one of them contains the resource, before or at the same time as requesting the resource from the reflector/origin server. If a peer cache responds positively in a limited period of time (preferably a small fraction of a second), the resource will be retrieved from the peer cache.

C4. The repeater then constructs a reply including the requested resource (which was retrieved from the cache or from the origin server) and sends that reply to the requesting client.

C5. Details about the transaction, such as the associated reflector, the current time, the address of the requester, the URL requested, and the type of response generated, are written to a local log file at the repeater.

Note that the bottom row of FIG. 2 refers to an origin server, or a reflector, or a repeater, depending on what the URL in step A1 identifies.

Selecting the Best Repeater

If the reflector 108 determines that it will reflect the request, it must then select the best repeater to handle that request (as referred to in step B3-1). This selection is performed by the Best Repeater Selector (BRS) mechanism described here.

The goal of the BRS is to select, quickly and heuristically, an appropriate repeater for a given client given only the network address of the client. An appropriate repeater is one which is not too heavily loaded and which is not too far from the client in terms of some measure of network distance. The mechanism used here relies on specific, compact, pre-computed data to make a fast decision. Other, dynamic solutions can also be used to select an appropriate repeater.

The BRS relies on three pre-computed tables, namely the Group Reduction Table, the Link Cost Table, and the Load Table. These three tables (described below) are computed off-line and downloaded to each reflector by its contact in the repeater network.

The Group Reduction Table places every network address into a group, with the goal that addresses in a group share relative costs, so that they would have the same best repeater under varying conditions (i.e., the BRS is invariant over the members of the group).

The Link Cost Table is a two dimensional matrix which specifies the current cost between each repeater and each group. Initially, the link cost between a repeater and a group is defined as the "normalized link cost" between the repeater and the group, as defined below. Over time, the table will be updated with measurements which more accurately reflect the relative cost of transmitting a file between the repeater and a member of the group. The format of the Link Cost Table is <Group ID><Group ID><link cost>, where the Group ID's are given as AS numbers.

The Load Table is a one dimensional table which identifies the current load at each repeater. Because repeaters may have different capacities, the load is a value that represents the ability of a given repeater to accept additional work. Each repeater sends its current load to a central master repeater at regular intervals, preferably at least approximately once a minute. The master repeater broadcasts the Load Table to each reflector in the network, via the contact repeater.

A reflector is provided entries in the Load Table only for repeaters which it is assigned to use. The assignment of repeaters to reflectors is performed centrally by a repeater network operator at the master repeater. This assignment makes it possible to modify the service level of a given reflector. For instance, a very active reflector may use many repeaters, whereas a relatively inactive reflector may use few repeaters.

Tables may also be configured to provide selective repeater service to subscribers in other ways, e.g., for their clients in specific geographic regions, such as Europe or Asia.

Measuring Load

In the presently preferred embodiments, repeater load is measured in two dimensions, namely

1. requests received by the repeater per time interval (RRPT), and
2. bytes sent by the repeater per time interval (BSPT).

For each of these dimensions, a maximum capacity setting is set. The maximum capacity indicates the point at which the repeater is considered to be fully loaded. A higher RRPT capacity generally indicates a faster processor, whereas a higher BSPT capacity generally indicates a wider network pipe. This form of load measurement assumes that a given server is dedicated to the task of repeating.

Each repeater regularly calculates its current RRPT and BSPT, by accumulating the number of requests received and bytes sent over a short time interval. These measurements are used to determine the repeater's load in each of these dimensions. If a repeater's load exceeds its configured capacity, an alarm message is sent to the repeater network administrator.

The two current load components are combined into a single value indicating overall current load. Similarly, the two maximum capacity components are combined into a single value indicating overall maximum capacity. The components are combined as follows:

current-load=B×current RRPT+(1−B)×current BSPT

max-load=B×max RRPT+(1−B)×max BSPT

The factor B, a value between 0 and 1, allows the relative weights of RRPT and BSPT to be adjusted, which favors consideration of either processing power or bandwidth.

The overall current load and overall maximum capacity values are periodically sent from each repeater to the master repeater, where they are aggregated in the Load Table, a table summarizing the overall load for all repeaters. Changes in the Load Table are distributed automatically to each reflector.

While the preferred embodiment uses a two-dimensional measure of repeater load, any other measure of load can be used.

Combining Link Costs and Load

The BRS computes the cost of servicing a given client from each eligible repeater. The cost is computed by combining the available capacity of the candidate repeater with the cost of the link between that repeater and the client. The link cost is computed by simply looking it up in the Link Cost table.

The cost is determined using the following formula:

threshold=K * max-load

capacity=max(max-load−current-load, ε)

capacity=min(capacity, threshold)

cost=link-cost * threshold/cacity

In this formula, ε is a very small number (epsilon) and K is a tuning factor initial set to 0.5. This formula causes the cost to a given repeater to be increased, at a rate defined by K, if its capacity falls below a configurable threshold.

Given the cost of each candidate repeater, the BRS selects all repeaters within a delta factor of the best score. From this set, the result is selected at random.

The delta factor prevents the BRS from repeatedly selecting a single repeater when scores are similar. It is generally required because available information about load and link costs loses accuracy over time. This factor is tunable.

Best Repeater Selector (BRS)

The BRS operates as follows, with reference to FIG. 6: Given a client network address and the three tables described above:

E1. Determine which group the client is in using the Group Reduction Table.

E2. For each repeater in the Link Cost Table and Load Table, determine that repeater's combined cost as follows:

E2a. Determine the maximum and current load on the repeater (using the Load Table).

E2b. Determine the link cost between the repeater and the client's group (using the Link Cost Table).

E2c. Determine the combined cost as described above.

E3. Select a small set of repeaters with the lowest cost.

E4. Select a random member from the set.

Preferably the results of the BRS processing are maintained in a local cache at the reflector **108**. Thus, if the best repeater has recently been determined for a given client (i.e., for a given network address), that best repeater can be reused quickly without being re-determined. Since the calculation described above is based on statically, pre-computed tables, if the tables have not changed then there is no need to re-determine the best repeater.

Determining the Group Reduction and Link Cost Tables

The Group Reduction Table and Link Cost Table used in BRS processing are created and regularly updated by an independent procedure referred to herein as NetMap. The NetMap procedure is run by executing several phases (described below) as needed.

The term Group is used here to refers to an IP "address group".

The term Repeater Group refers to a Group that contains the IP address of a repeater.

The term link cost refers to a statically determined cost for transmitting data between two Groups. In a presently preferred implementation, this is the minimum of the sums of the costs of the links along each path between them. The link costs of primary concern here are link costs between a Group and a Repeater Group.

The term relative link cost refers to the link cost relative to other link costs for the same Group which is calculated by subtracting the minimum link cost from a Group to any Repeater Group from each of its link costs to a Repeater Group. The term Cost Set refers to a set of Groups that are equivalent in regard to Best Repeater Selection. That is, given the information available, the same repeater would be selected for any of them.

The NetMap procedure first processes input files to create an internal database called the Group Registry. These input files describe groups, the IP addresses within groups, and links between groups, and come a variety of sources, including publicly available Internet Routing Registry (IRR) databases, BGP router tables, and probe services that are located at various points around the Internet and use publicly available tools (such as "traceroute") to sample data paths. Once this processing is complete, the Group Registry contains essential information used for further processing, namely (1) the identity of each group, (2) the set of IP addresses in a given group, (3) the presence of links between groups indicating paths over which information may travel, and (4) the cost of sending data over a given link.

The following processes are then performed on the Group Registry file.

Calculate Repeater Group link costs

The NetMap procedure calculates a "link cost" for transmission of data between each Repeater Group and each Group in the Group Registry. This overall link cost is defined as the minimum cost of any path between the two groups, where the cost of a path is equal to the sum of the costs of the individual links in the path. The link cost algorithm presented below is essentially the same as algorithm #562 from ACM journal Transactions on Mathematical Software: "Shortest Path From a Specific Node to All Other Nodes in a Network" by U. Pape, ACM TOMS 6 (1980) pp. 450–455, http://www.netlib.org/toms/562.

In this processing, the term Repeater Group refers to a Group that contains the IP address of a repeater. A group is a neighbor of another group if the Group Registry indicates that there is a link between the two groups.

```
For each target Repeater Group T:
  Initialize the link cost between T and itself to zero.
  Initialize the link cost between T and every other Group
    to infinity.
  Create a list L that will contain Groups that are equidistant
    from the target Repeater Group T.
  Initialize the list L to contain just the target Repeater
    Group T itself.
  While the list L is not empty:
    Create an empty list L' of neighbors of members of the
      list L.
    For each Group G in the list L:
      For each Group N that is a neighbor of G:
        Let cost refer to the sum of the link cost between
          T and G, and the link cost between G and N.
          The cost between T and G was determined in
          the previous pass of the algorithm; the link cost
          between G and N is from the Group Registry.
        If cost is less than the link cost between T and N:
          Set the link cost between T and N to cost.
          Add N to L' if it is not already on it.
    Set L to L'.
```

Calculate Cost Sets

A Cost Set is a set of Groups that are equivalent with respect to Best Repeater Selection. That is, given the information available, the same repeater would be selected for any of them.

The "cost profile" of a Group G is defined herein as the set of costs between G and each Repeater. Two cost profiles are said to be equivalent if the values in one profile differ from the corresponding values in the other profile by a constant amount.

Once a client Group is known, the Best Repeater Selection algorithm relies on the cost profile for information about the Group. If two cost profiles are equivalent, the BRS algorithm would select the same repeater given either profile.

A Cost Set is then a set of groups that have equivalent cost profiles.

The effectiveness of this method can be seen, for example, in the case where all paths to a Repeater from some Group A pass through some other Group B. The two Groups have equivalent cost profiles (and are therefore in the same Cost Set) since whatever Repeater is best for Group A is also going to be best for Group B, regardless of what path is taken between the two Groups.

By normalizing cost profiles, equivalent cost profiles can be made identical. A normalized cost profile is a cost profile in which the minimum cost has the value zero. A normalized cost profile is computed by finding the minimum cost in the profile, and subtracting that value from each cost in the profile.

Cost Sets are then computed using the following algorithm:

```
For each Group G:
  Calculate the normalized cost profile for G
  Look for a Cost Set with the same normalized cost
    profile.
  If such as set is found, add G to the existing Cost Set;
  otherwise, create a new Cost Set with the calculated
    normalized cost profile, containing only G.
```

The algorithm for finding Cost Sets employs a hash table to reduce the time necessary to determine whether the

desired Cost Set already exists. The hash tahle uses a hash value computed from cost profile of G.

Each Cost Set is then numhered with a unique Cost Sent Index number. Cost Sets are then used in a straightforward manner to generate the Link Cost Tahle, which gives the cost from-each-Cost-Set-to-each-Repeater.

As described below, the Group Reduction Table maps every IP address to one of these Cost Sets.

Build IP Map

The IP Map is a sorted list of records which map IP address ranges to Link Cost Table keys. The format of the IP map is:

<base IP address><max IP address><Link Cost Table key>

where IP addresses are presently represented by 32-hit integers. The entries are sorted by descending base address, and hy ascending maximum address among equal base addresses, and by ascending Link Cost Table key among equal hase addresses and maximum addresses. Note that ranges may overlap.

The NetMap procedure generates an intermediate IP map containing a map between IP address ranges and Cost Set numbers as follows:

```
For each Cost Set S:
  For each Group G in S:
    For each IP address range in G:
      Add a triple (low address, high address, Cost Set
        numher of S) to the IP map.
```

The IP map file is then sorted hy descending hase address, and hy ascending maximum address among equal hase addresses, and by ascending Cost Set number among equal base addresses and maximum addresses. The sort order for the hase address and maximum address minimizes the time to build the Group Reduction Table and produces the proper results for overlapping entries.

Finally, the NetMap procedure creates the Group Reduction Tahle hy processing the sorted IP map. The Group Reduction Tahle maps IP addresses (specified hy ranges) into Cost Set numbers. Special processing of the IP map file is required in order to detect overlapping address ranges, and to merge adjacent address ranges in order to minimize the size of the Group Reduction Table.

An ordered list of address range segments is maintained, each segment consisting of a hase address B and a Cost Set number N, sorted hy hase address B. (The maximum address of a segment is the base address of the next segment minus one.)

The following algorithm is used:

```
Initialize the list with the elements [-infinity,
  NOGROUP], [+infinity, NOGROUP].
  For each entry in the IP map, in sorted order, consisting
    of (b, m, s),
    Insert (b, m, s) in the list (recall that IP map entries
      are of the form (low address, high address Cost Set
      number of S))
  For each reserved LAN address range (b, m): Insert (b,
    m, LOCAL) in the list.
  For each Repeater at address a:
    Insert (a, a, REPEATER) in the list.
  For each segment S in the ordered list:
    Merge S with following segments with the same
      Cost Set
    Create a Group Reduction Table entry with base
      address from the hase address of S,
       max address=next segment's base-1,
       group ID=Cost Set number of S.
```

A reserved LAN address range is an address range reserved for use hy LANs which should not appear as a glohal Internet address. LOCAL is a special Cost Set index different from all others, indicating that the range maps to a client which should never be reflected. REPEATER is a special Cost Set index different from all others, indicating that the address range maps to a repeater. NOGROUP is a special Cost Set index different from all others, indicating that this range of addresses has no known mapping.

```
Given (B, M, N), insert an entry in the ordered address list
  as follows:
  Find the last segment (AB, AN) for which AB is less than
    or equal to B.
  If AB is less than B, insert a new segment (B, N) after
    (AB, AN).
  Find the last segment (YB, YN) for which YB is less than
    or equal to M.
  Replace by (XB, N) any segment (XB, NOGROUP) for
    which XB is greater than B and less than YB.
  If YN is not N, and either YN is NOGROUP or YB is less
    than or equal to B,
    Let (ZB, ZN) he the segment following (YB, YN).
    If M+1 is less than ZB, insert a new segment (M+1,
      YN) before (ZB, ZN).
    Replace (YB, YN) by (YB, N).
```

Rewriting HTML Resources

As explained above with reference to FIG. 3 (B5), when a reflector or repeater serves a resource which itself includes resource identifiers (e.g., a HTML resource), that resource is modified (rewritten) to pre-reflect resource identifiers (URLs) of repeatahle resources that appear in the resource. Rewriting ensures that when a hrowser requests repeatable resources identified hy the requested resource, it gets them from a repeater without going hack to the origin server, hut when it requests non-repeatable resources identified by the requested resource, it will go directly to the origin server. Without this optimization, the browser would either make all requests at the origin server (increasing traffic at the origin server and necessitating far more redirections from the origin server), or it would make all requests at the repeater (causing the repeater to redundantly request and copy resources which could not be cached, increasing the overhead of serving such resources).

Rewriting requires that a repeater has heen selected (as described ahove with reference to the Best Repeater Selector). Rewriting uses a so-called BASE directive. The BASE directive lets the HTML identify a different base server. (The hase address is normally the address of the HTML resource.)

Rewriting is performed as follows:

F1. A BASE directive is added at the beginning of the HTML resource, or modified where necessary. Normally, a browser interprets relative URLs as heing relative to the default base address, namely, the URL of the HTML resource (page) in which they are encountered. The BASE address added specifies the resource at the reflector which originally served the resource. This means that unprocessed relative URLs (such as those generated by Javascript™ programs) will be interpreted as relative to the reflector. Without this BASE address, hrowsers would combine relative addresses with repeater names to create URLs which were not in the form required by repeaters (as described above in step D1).

F2. The rewriter identifies directives, such as emhedded images and anchors, containing URLs. If the rewriter is

running in a reflector, it must parse the HTML file to identify these directives. If it is running in a repeater, the rewriter may have access to pre-computed information that identifies the location of each URL (placed in the HTML file in step F4).

F3. For each URL encountered in the resource to be re-written, the rewriter must determine whether the URL is repeatable (as in steps B1–B2). If the URL is not repeatable, it is not modified. On the other hand, if the URL is repeatable, it is modified to refer to the selected repeater.

F4. After all URLs have been identified and modified, if the resource is being served to a repeater, a table is appended at the beginning of the resource that identifies the location and content of each URL encountered in the resource. (This step is an optimization which eliminates the need for parsing HTML resources at the repeater.)

F5. Once all changes have been identified, a new length is computed for the resource (page). The length is inserted in the HTTP header prior to serving the resource.

An extension of HTML, known as XML, is currently being developed. The process of rewriting URLs will be similar for XML, with some differences in the mechanism that parses the resource and identifies embedded URLs.

Handling Non-HTTP Protocols

This invention makes it possible to reflect references to resources that are served by protocols other than HTTP, for instance, the File Transfer Protocol (FTP) and audio/video stream protocols. However, many protocols do not provide the ability to redirect requests. It is, however, possible to redirect references before requests are actually made by rewriting URLs embedded in HTML pages. The following modifications to the above algorithms are used to support this capability.

In F4, the rewriter rewrites URLs for servers if those servers appear in a configurable table of cooperating origin server or so-called co-servers. The reflector operator can define this table to include FTP servers and other servers. A rewritten URL that refers to a non-HTTP resource takes the form:

http://<repeater>/<origin server>@proxy=<scheme>]:<type>]@/resource

where <scheme> is a supported protocol name such as "ftp". This URL format is an alternative to the form shown in B3.

In C3, the repeater looks for a protocol embedded in the arriving request. If a protocol is present and the requested resource is not already cached, the repeater uses the selected protocol instead of the default HTTP protocol to request the resource when serving it and storing it in the cache.

System Configuration and Management

In addition to the processing described above, the repeater network requires various mechanisms for system configuration and network management. Some of these mechanisms are described here.

Reflectors allow their operators to synchronize repeater caches by performing publishing operations. The process of keeping repeater caches synchronized is described below. Publishing indicates that a resource or collection of resources has changed.

Repeaters and reflectors participate in various types of log processing. The results of logs collected at repeaters are collected and merged with logs collected at reflectors, as described below.

Adding Subscribers to the Repeater Network

When a new subscriber is added to the network, information about the subscriber is entered in a Subscriber Table at the master repeater and propagated to all repeaters in the network. This information includes the Committed Aggregate Information Rate (CAIR) for servers belonging to the subscriber, and a list of the repeaters that may be used by servers belonging to the subscriber.

Adding Reflectors to the Repeater Network

When a new reflector is added to the network, it simply connects to and announces itself to a contact repeater, preferably using a securely encrypted certificate including the repeater's subscriber identifier.

The contact repeater determines whether the reflector network address is permitted for this subscriber. If it is, the contact repeater accepts the connection and updates the reflector with all necessary tables (using version numbers to determine which tables are out of date).

The reflector processes requests during this time, but is not "enabled" (allowed to reflect requests) until all of its tables are current.

Keeping Repeater Caches Synchronized

Repeater caches are coherent, in the sense that when a change to a resource is identified by a reflector, all repeater caches are notified, and accept the change in a single transaction.

Only the identifier of the changed resource (and not the entire resource) is transmitted to the repeaters; the identifier is used to effectively invalidate the corresponding cached resource at the repeater. This process is far more efficient than broadcasting the content of the changed resource to each repeater.

A repeater will load the newly modified resource the next time it is requested.

A resource change is identified at the reflector either manually by the operator, or through a script when files are installed on the server, or automatically through a change detection mechanism (e.g., a separate process that checks regularly for changes).

A resource change causes the reflector to send an "invalidate" message to its contact repeater, which forwards the message to the master repeater. The invalidate message contains a list of resource identifiers (or regular expressions identifying patterns of resource identifiers) that have changed. (Regular expressions are used to invalidate a directory or an entire server.) The repeater network uses a two-phase commit process to ensure that all repeaters correctly invalidate a given resource.

The invalidation process operates as follows:

The master broadcasts a "phase 1" invalidation request to all repeaters indicating the resources and regular expressions describing sets of resources to be invalidated.

When each repeater receives the phase 1 message, it first places the resource identifiers or regular expressions into a list of resource identifiers pending invalidation.

Any resource requested (in C3) that is in the pending invalidation list may not be served from the cache. This prevents the cache from requesting the resource from a peer cache which may not have received an invalidation notice. Were it to request a resource in this manner, it might replace the newly invalidated resource by the same, now stale, data.

The repeater then compares the resource identifier of each resource in its cache against the resource identifiers and regular expressions in the list.

Each match is invalidated by marking it stale and optionally removing it from the cache. This means that a future request for the resource will cause it to retrieve a new copy of the resource from the reflector.

When the repeater has completed the invalidation, it returns an acknowledgment to the master. The master waits until all repeaters have acknowledged the invalidation request.

If a repeater fails to acknowledge within a given period, it is disconnected from the master repeater. When it reconnects, it will be told to flush its entire cache, which will eliminate any consistency problem. (To avoid flushing the entire cacbe, the master could keep a log of all invalidations performed, sorted by date, and flush only files invalidated since the last time the reconnecting repeater successfully completed an invalidation. In tbe presently preferred embodiments this is not done since it is believed that repeaters will seldom disconnect.)

When all repeaters have acknowledged invalidation (or timed out) the repeater broadcasts a "phase 2" invalidation request to all repeaters. This causes the repeaters to remove the corresponding resource identifiers and regular expressions from tbe list of resource identifiers pending invalidation.

In another embodiment, the invalidation request will be extended to allow a "server push". In sucb requests, after phase 2 of the invalidation process bas completed, the repeater receiving the invalidation request will immediately request a new copy of the invalidated resource to place in its cacbe.

Logs and Log Processing

Web server activity logs are fundamental to monitoring the activity in a Web site. This invention creates "merged logs" tbat combine the activity at reflectors witb the activity at repeaters, so tbat a single activity log appears at tbe origin server showing all Web resource requests made on bebalf of that site at any repeater.

This merged log can be processed by standard processing tools, as if it had been generated locally.

On a periodic basis, tbe master repeater (or its delegate) collects logs from each repeater. The logs collected are merged, sorted by reflector identifier and timestamp, and stored in a dated file on a per-reflector basis. The merged log for a given reflector represents the activity of all repeaters on bebalf of that reflector. On a periodic basis, as configured by the reflector operator, a reflector contacts the master repeater to request its merged logs. It downloads tbese and merges them with its locally maintained logs, sorting by timestamp. The result is a merged log tbat represents all activity on behalf of repeaters and the given reflector.

Activity logs are optionally extended with information important to the repeater network, if the reflector is configured to do so by the reflector operator. In particular, an "extended status code" indicates information about each request, such as:

1. request was served by a reflector locally;
2. request was reflected to a repeater;*
3. request was served by a reflector to a repeater;*
4. request for non-repeatable resource was served by repeater;*
5. request was served by a repeater from the cache;
6. request was served by a repeater after filling cacbe;
7. request pending invalidation was served by a repeater.

(The activities marked with "*" represent intermediate states of a request and do not normally appear in a final activity log.)

In addition, activity logs contain a duration, and extended precision timestamps. The duration makes it possible to analyze the time required to serve a resource, the bandwidtb used, tbe number of requests handled in parallel at a given time, and other quite useful information. The extended precision timestamp makes it possible to accurately merge activity logs.

Repeaters use the Network Time Protocol (NTP) to maintain synchronized clocks. Reflectors may either use NTP or calculate a time bias to provide roughly accurate timestamps relative to tbeir contact repeater.

Enforcing Committed Aggregate Information Rate

The repeater network monitors and limits the aggregate rate at whicb data is served on behalf of a given subscriber by all repeaters. This mechanism provides the following benefits:

1. provides a means of pricing repeater service;
2. provides a means for estimating and reserving capacity at repeaters;
3. provides a means for preventing clients of a busy site from limiting access to other sites.

For each subscriber, a "thresbold aggregate information rate" (TAIR) is configured and maintained at the master repeater. This threshold is not necessarily the committed rate, it may be a multiple of committed rate, based on a pricing policy.

Each repeater measures the information rate component of each reflector for which it serves resources, periodically (typically about once a minute), by recording the number of bytes transmitted on bebalf of tbat reflector each time a request is delivered. The table thus created is sent to the master repeater once per period. The master repeater combines tbe tables from eacb repeater, summing the measured information of each reflector over all repeaters tbat serve resources for tbat reflector, to determine the "measured aggregate information rate" (MAIR) for each reflector.

If tbe MAIR for a given reflector is greater than the TAIR for tbat reflector, the MAIR is transmitted by the master to all repeaters and to the respective reflector.

When a reflector receives a request, it determines whether its most recently calculated MAIR is greater tban its TAIR. If this is the case, the reflector probabilistically decides wbether to suppress reflection, by serving the request locally (in B2). The probability of suppressing the reflection increases as an exponential function of the difference between the MAIR and the CAIR.

Serving a request locally during a peak period may strain the local origin server, but it prevents this subscriber from taking more than allocated bandwidth from the shared repeater network.

When a repeater receives a request for a given subscriber (in C2), it determines whether the subscriber is running near its tbreshold aggregate information rate. If this is the case, it probabilistically decides whether to reduce its load by redirecting the request back to the reflector. The probability increases exponentially as the reflector's aggregate information rate approacbes its limit.

If a request is reflected back to a reflector, a special cbaracter string is attached to the resource identifier so that the receiving reflector will not attempt to reflect it again. In the current system, this string has the form

"src=overload".

The reflector tests for this string in B2.

The mechanism for limiting Aggregate Information Rate described above is fairly coarse. It limits at the level of sessions with clients (since once a client has been reflected to a given repeater, the rewriting process tends to keep the client coming back to that repeater) and, at best, individual requests for resources. A more fine-grained mechanism for

enforcing TAIR limits within repeaters operates by reducing the bandwidth consumption of a busy subscriber when other subscribers are competing for bandwidth.

The fine-grained mechanism is a form of data "rate shaping". It extends the mechanism that copies resource data to a connection when a reply is being sent to a client. When an output channel is established at the time a request is received, the repeater identifies which subscriber the channel is operating for, in C2, and records the subscriber in a data field associated with the channel. Each time a "write" operation is about to be made to the channel, the Metered Output Stream first inspects the current values of the MAIR and TAIR, calculated above, for the given subscriber. If the MAIR is larger than the TAIR, then the mechanism pauses briefly before performing the write operation. The length of the pause is proportional to the amount the MAIR exceeds the TAIR. The pause ensures that tasks sending other resources to other clients, perhaps on behalf of other subscribers, will have an opportunity to send their data.

Repeater Network Resilience

The repeater network is capable of recovering when a repeater or network connection fails.

A repeater cannot operate unless it is connected to the master repeater. The master repeater exchanges critical information with other repeaters, including information about repeater load, aggregate information rate, subscribers, and link cost.

If a master fails, a "succession" process ensures that another repeater will take over the role of master, and the network as a whole will remain operational. If a master fails, or a connection to a master fails through a network problem, any repeater attempting to communicate with the master will detect the failure, either through an indication from TCP/IP, or by timing out from a regular "heartbeat" message it sends to the master.

When any repeater is disconnected from its master, it immediately tries to reconnect to a series of potential masters based on a configurable file called its "succession list".

The repeater tries each system on the list in succession until it successfully connects to a master. If in this process, it comes to its own name, it takes on the role of master, and accepts connections from other repeaters. If a repeater which is not at the top of the list becomes the master, it is called the "temporary master".

A network partition may cause two groups of repeaters each to elect a master. When the partition is corrected, it is necessary that the more senior master take over the network. Therefore, when a repeater is temporary master, it regularly tries to reconnect to any master above it in the succession list. If it succeeds, it immediately disconnects from all of the repeaters connected to it. When they retry their succession lists, they will connect to the more senior master repeater.

To prevent losses of data, a temporary master does not accept configuration changes and does not process log files. In order to take on these tasks, it must be informed that it is primary master by manual modification of its successor list. Each repeater regularly reloads its successor list to determine whether it should change its idea of who the master is.

If a repeater is disconnected from the master, it must resynchronize its cache when it reconnects to the master. The master can maintain a list of recent cache invalidations and send to the repeater any invalidations it was not able to process while disconnected. If this list is not available for some reason (for instance, because the reflector has been disconnected too long), the reflector must invalidate its entire cache.

A reflector is not permitted to reflect requests unless it is connected to a repeater. The reflector relies on its contact repeater for critical information, such as load and Link Cost Tables, and current aggregate information rate. A reflector that is not connected to a repeater can continue to receive requests and handle them locally.

If a reflector loses its connection with a repeater, due to a repeater failure or network outage, it continues to operate while it tries to connect to a repeater.

Each time a reflector attempts to connect to a repeater, it uses DNS to identify a set of candidate repeaters given a domain name that represents the repeater network. The reflector tries each repeater in this set until it makes a successful contact. Until a successful contact is made, the reflector serves all requests locally. When a reflector connects to a repeater, the repeater can tell it to attempt to contact a different repeater; this allows the repeater network to ensure that no individual repeater has too many contacts.

When contact is made, the reflector provides the version number of each of its tables to its contact repeater. The repeater then decides which tables should be updated and sends appropriate updates to the reflector. Once all tables have been updated, the repeater notifies the reflector that it may now start reflecting requests.

Using a Proxy Cache within a Repeater

Repeaters are intentionally designed so that any proxy cache can be used as a component within them. This is possible because the repeater receives HTTP requests and converts them to a form recognized by the proxy cache.

On the other hand, several modifications to a standard proxy cache have been or may be made as optimizations. This includes, in particular, the ability to conveniently invalidate a resource, the ability to support cache quotas, and the ability to avoid making an extra copy of each resource as it passes from the proxy cache through the repeater to the requester.

In a preferred embodiment, a proxy cache is used to implement C3. The proxy cache is dedicated for use only by one or more repeaters. Each repeater requiring a resource from the proxy cache constructs a proxy request from the inbound resource request. A normal HTTP GET request to a server contains only the pathname part of the URL—the scheme and server name are implicit. (In an HTTP GET request to a repeater, the pathname part of the URL includes the name of the origin server on behalf of which the request is being made, as described above.) However, a proxy agent GET request takes an entire URL. Therefore, the repeater must construct a proxy request containing the entire URL from the path portion of the URL it receives. Specifically, if the incoming request takes the form:

GET/<origin server>/<path>

then the repeater constructs a proxy request of the form:

GET http://<origin server>/<path>

and if the incoming request takes the form:

GET<origin server>@proxy=<scheme>:<type>@/<path>

then the repeater constructs a proxy request of the form:

GET<scheme>://<origin server>/<path>

Cache Control

HTTP replies contain directives called cache control directives, which are used to indicate to a cache whether the attached resource may be cached and if so, when it should

expire. A Web site administrator configures the Web site to attach appropriate directives. Often, the administrator will not know how long a page will he fresh, and must define a short expiration time to try to prevent caches from serving stale data. In many cases, a Web site operator will indicate a short expiration time only in order to receive the requests (or hits) that would otherwise he masked by the presence of a cache. This is known in the industry as "cache-busting". Although some cache operators may consider cache-busting to be impolite, advertisers who rely on this information may consider it imperative.

When a resource is stored in a repeater, its cache directives can be ignored by the repeater, hecause the repeater receives explicit invalidation events to determine when a resource is stale. When a proxy cache is used as the cache at the repeater, the associated cache directives may be temporarily disahled. However, they must be re-enahled when the resource is served from the cache to a client, in order to permit the cache-control policy (including any cache-busting) to take effect.

The present invention contains mechanisms to prevent the proxy cache within a repeater from honoring cache control directives, while permitting the directives to be served from the repeater.

When a reflector serves a resource to a repeater in B4, it replaces all cache directives by modified directives that are ignored by the repeater proxy cache. It does this by prefixing a distinctive string such as "wr-" to the beginning of the HTTP tag. Thus, "expires" becomes "wr-expires", and "cache-control" hecomes "wr-cache-control". This prevents the proxy cache itself from honoring the directives. When a repeater serves a resource in C4, and the requesting client is not another repeater, it searches for HTTP tags beginning with "wr-" and removes the "wr-". This allows the clients retrieving the resource to honor the directives.

Resource Revalidation

There are several cases where a resource may be cached so long as the origin server is consulted each time it is served. In one case, the request for the resource is attached to a so-called "cookie". The origin server must be presented with the cookie to record the request and determine whether the cached resource may he served or not. In another case, the request for the resource is attached to an authentication header (which identifies the requester with a user id and password). Each new request for the resource must he tested at the origin server to assure that the requester is authorized to access the resource.

The HTTP 1.1 specification defines a reply header titled "Must-Revalidate" which allows an origin server to instruct a proxy cache to "revalidate" a resource each time a request is received. Normally, this mechanism is used to determine whether a resource is still fresh. In the present invention, Must-Revalidate makes it possible to ask an origin server to validate a request that is otherwise served from a repeater.

The reflector rule hase contains information that determines which resources may he repeated but must be revalidated each time they are served. For each such resource, in B4, the reflector attaches a Must-Revalidate header. Each time a request comes to a repeater for a cached resource marked with a Must-Revalidate header, the request is forwarded to the reflector for validation prior to serving the requested resource.

Cache Quotas

The cache component of a repeater is shared among those subscribers that reflect clients to that repeater. In order to allow suhscribers fair access to storage facilities, the cache may be extended to support quotas.

Normally, a proxy cache may be configured with a disk space threshold T. Whenever more than T bytes are stored in the cache, the cache attempts to find resources to eliminate.

Typically a cache uses the least-recently-used (LRU) algorithm to determine which resources to eliminate; more sophisticated caches use other algorithms. A cache may also support several threshold values-for instance, a lower threshold which, when reached, causes a low priority background process to remove items from the cache, and a higher threshold which, when reached, prevents resources from heing cached until sufficient free disk space has been reclaimed.

If two subscribers A and B share a cache, and more resources of subscriber A are accessed during a period of time than resources of subscriber B, then fewer of B's resources will be in the cache when new requests arrive. It is possihle that, due to the behavior of A, B's resources will never be cached when they are requested. In the present invention, this behavior is undesirable. To address this issue, the invention extends the cache at a repeater to support cache quotas.

The cache records the amount of space used by each suhscriber in D_S, and supports a configurable threshold T_S for each subscriber.

Whenever a resource is added to the cache (at C3), the value D_S is updated for the subscriber providing the resource. If D_S is larger than T_S, the cache attempts to find resources to eliminate, from among those resources associated with subscriber S. The cache is effectively partitioned into separate areas for each subscriher.

The original threshold T is still supported. If the sum of reserved segments for each suhscriber is smaller than the total space reserved in the cache, the remaining area is "common" and subject to competition among subscribers.

Note, this mechanism might be implemented hy modifying the existing proxy cache discussed above, or it might also be implemented without modifying the proxy cache—if the proxy cache at least makes it possible for an external program to obtain a list of resources in the cache, and to remove a given resource from the cache.

Rewriting from Repeaters

When a repeater receives a request for a resource, its proxy cache may be configured to determine whether a peer cache contains the requested resource. If so, the proxy cache obtains the resource from the peer cache, which can be faster than ohtaining it from the origin server (the reflector). However, a consequence of this is that rewritten HTML resources retrieved from the peer cache would identify the wrong repeater. Thus, to allow for cooperating proxy caches, resources are preferahly rewritten at the repeater.

When a resource is rewritten for a repeater, a special tag is placed at the beginning of the resource. When constructing a reply, the repeater inspects the tag to determine whether the resource indicates that additional rewriting is necessary. If so, the repeater modifies the resource hy replacing references to the old repeater with references to the new repeater.

It is only necessary to perform this rewriting when a resource is served to the proxy cache at another repeater.

Repeater-Side Include

Sometimes, an origin server constructs a custom resource for each request (for instance, when inserting an advertisement hased on the history of the requesting client). In such a case, that resource must be served locally rather than repeated. Generally, a custom resource contains, along with the custom information, text and references to other, repeatable, resources.

The process that assembles a "page" from a text resource and possibly one or more image resources is performed by the Web browser, directed by HTML. However, it is not possible using HTML to cause a browser to assemble a page using text or directives from a separate resource. Therefore, custom resources often necessarily contain large amounts of static text that would otherwise be repeatable.

To resolve this potential inefficiency, repeaters recognize a special directive called a "repeater side include". This directive makes it possible for the repeater to assemble a custom resource, using a combination of repeatable and local resources. In this way, the static text can be made repeatable, and only the special directive need be served locally by the reflector.

For example, a resource X might consist of custom directives selecting an advertising banner, followed by a large text article. To make this resource repeatable, the Web site administrator must break out a second resource, Y, to select the banner. Resource X is modified to contain a repeater-side include directive identifying resource Y, along with the article. Resource Y is created and contains only the custom directives selecting an ad banner. Now resource X is repeatable, and only resource Y, which is relatively small, is not repeatable.

When a repeater constructs a reply, it determines whether the resource being served is an HTML resource, and if so, scans it for repeater-side include directives. Each such directive includes a URL, which the repeater resolves and substitutes in place of the directive. The entire resource must be assembled before it is served, in order to determine its final size, as the size is included in a reply header ahead of the resource.

Thus, a method and apparatus for dynamically replicating selected resources in computer networks is provided. One skilled in the art will appreciate that the present invention can be practiced by other than the described embodiments, which are presented for purposes of illustration and not limitation, and the present invention is limited only by the claims that follow.

What is claimed:

1. A method of processing resource requests in a computer network, the method comprising,

(i) by a client:

(A) making a request for a particular resource from an origin server, the request including a resource identifier for the particular resource;

(ii) by a reflector:

(B) intercepting the request from the client to the origin server;

(C) selecting a repeater to process the request, wherein the repeater is selected based on a predicted cost or speed of transmission between the repeater and the client;

(D) providing to the client a modified resource identifier designating the repeater;

(iii) by the client:

(E) receiving the modified resource identifier from the reflector; and

(F) making a request for the particular resource from the repeater designated in the modified resource identifier;

(iv) by the repeater:

(G) receiving the request from the client; and

(H) returning the requested resource to the client.

2. A method as in claim 1 further comprising, by the repeater:

(I) making a request for the resource from the origin server; and

(J) receiving the resource from the origin server.

3. A method as in claim 1 wherein the selecting of a repeater by the reflector comprises:

(C1) partitioning the network into groups;

(C2) determining which group the client is in;

(C3) selecting, from a plurality of repeaters in the network, a set of repeaters having a lowest cost relative to the group which the client is in; and

(C4) selecting as the repeater a member of the selected set of repeaters.

4. A method as in claim 3, wherein the cost of a repeater is a value based on that repeater's current load and a maximum load for that repeater.

5. A method as in claim 3, wherein the cost of a repeater is a value based on a predicted cost or speed of transmission between the repeater and a client in the group.

6. A method as in claim 1 wherein the particular resource itself contains at least one other resource identifier of at least one other resource, the method further comprising:

rewriting the particular resource to replace at least some of the resource identifiers contained therein with modified resource identifiers designating a repeater instead of the origin server.

7. A method as in claim 6 wherein the rewriting is performed by one of the repeater, the reflector or another repeater.

8. A method of processing resource requests in a computer network, the method comprising,

(i) by a client:

(A) making a request for a particular resource from an origin server, the request including a resource identifier for the particular resource;

(ii) by a reflector:

(B) intercepting the request from the client to the origin server;

(C) determining whether to reflect the request to a repeater;

(D) when the reflector determines not to reflect the request, forwarding the request to the origin server, otherwise

(D1) selecting a repeater to process the request, wherein the repeater is selected based on a predicted cost or speed of transmission between the repeater and the client;

(D2) providing to the client a modified resource identifier designating the repeater.

9. A method as in claim 8, further comprising, when the reflector determines to reflect the request,

(iii) by the client:

(E) receiving the modified resource identifier from the reflector; and

(F) making a request for the particular resource from the repeater designated in the modified resource identifier;

(iv) by the repeater:

(G) receiving the request from the client; and

(H) returning the requested resource to the client.

10. A method as in claim 8 wherein the reflector determines whether to reflect a request by comparing the resource identifier with regular expression patterns of repeatable resources.

11. A method as in claim 8, wherein the reflector has a threshold aggregate information rate (TAIR) associated therewith, and wherein the determining of whether to reflect the request to a repeater comprises:

determining whether the TAIR of the reflector is exceeded by a measured aggregate information rate (MAIR) for

the reflector, wherein the reflector determines not to reflect the request when the MAIR exceeds the TAIR for the reflector.

12. A method as in claim 8, wherein the reflector has a threshold aggregate information rate (TAIR) associated therewith, and wherein the determining of whether to reflect the request to a repeater comprises:

probabilistically determining whether the TAIR of the reflector is exceeded by a measured aggregate information rate (MAIR) for the reflector, wherein the reflector determines not to reflect the request as an exponential function of the difference between the MAIR and the TAIR.

13. A method as in any of claims 11–12, wherein the MAIR is obtained from repeaters according to the rate at which they have transmitted data on behalf of the reflector during a given time interval.

14. A method as in any one of claims 1–12 wherein the network is the Internet and wherein the resource identifier is a uniform resource locator (URL) for designating resources on the Internet, and wherein the modified resource identifier is a URL designating the repeater and indicating the reflector or origin server, and wherein the modified resource identifier is provided to the client using a REDIRECT message.

15. In a computer network wherein clients request resources from origin servers, a method comprising:

providing at least one repeater;

providing reflectors at some of the origin servers, each reflector intercepting client resource requests made to its respective origin server; and

each reflector selectively redirecting client resource requests for certain resources to one of the repeaters, wherein a reflector determines whether or not to redirect a client resource to a repeater based on a predicted cost or speed of transmission between the repeater and the client making the resource request.

16. A method as in claim 15 further comprising, by repeaters in the network:

servicing redirected client resource requests; and

selectively maintaining copies of requested resources,

whereby resources corresponding to redirected resource requests are selectively migrated from their origin servers to one or more repeaters.

17. A computer network comprising:

a plurality of origin servers, at least some of the origin servers having reflectors associated therewith;

a plurality of repeaters; and

a plurality of clients,

wherein each reflector is adapted to intercept resource requests made to its respective origin server and to selectively redirect the resource requests to a dynamically selected repeater, wherein a repeater is selected based on a predicted cost or speed of transmission between the repeater and the client making the resource request.

18. In a computer network wherein clients request resources from origin servers, a reflector mechanism associated with an origin server, the reflector mechanism comprising:

means for intercepting a resource request made by client of an origin server;

means for analyzing the resource request to determine whether to service the request locally at the origin server;

means for determining a best repeater in the network to service the request when the analyzing means determines that the request should not be serviced locally; and

means for redirecting the client to the best repeater, wherein a repeater is selected based on a predicted cost or speed of transmission between the repeater and the client making the resource request.

19. A reflector mechanism as in claim 18 wherein the network is partitioned into groups and the means for determining the best repeater comprises:

means for determining which group the client is in;

means for selecting, from a plurality of repeaters in the network, a set of repeaters having a lowest cost relative to the group the client is in; and

means for selecting as the best repeater a member of the set of repeaters.

20. A reflector mechanism as in claim 19, wherein the cost of a repeater is a value based on a predicted cost or speed of transmission between the repeater and a client in the group.

21. A mechanism as in claim 19, wherein the cost of a repeater is a value based on that repeaters current load and a maximum load for that repeater.

22. A reflector as in claim 18 wherein the resource itself contains resource identifiers, the repeater further comprising:

means for rewriting the resource to replace at least some of the resource identifiers contained therein with modified resource identifiers designating the repeater instead of the origin server.

23. A reflector as in claim 18 wherein the resource itself contains resource identifiers, the reflector further comprising:

means for rewriting the resource to replace at least some of the resource identifiers contained therein with modified resource identifiers designating the best repeater instead of the origin server.

24. In a computer network wherein clients request resources from origin servers, a repeater mechanism comprising:

means for receiving a resource request from a client;

means for determining whether the resource is available locally;

means for, when it is determined that the resource is not available locally, obtaining the resource from an origin server, wherein the origin server is selected based on a predicted cost or speed of transmission between the origin server and the client; and

means for providing the resource to the client.

25. A method of processing resource requests in a computer network, the method comprising,

by a client, making a request for a particular resource from an origin server, the request including a resource identifier for the particular resource, and wherein the particular resource itself contains at least one other resource identifier of at least one other resource;

a reflector intercepting the request from the client to the origin server;

selecting a repeater to process the request, wherein the repeater is selected based on a predicted cost or speed of transmission between the repeater and the client;

rewriting the particular resource to replace at least some of the resource identifiers contained therein with modified resource identifiers designating a repeater instead of the origin server;

providing to the client a modified resource identifier designating the repeater;

the client receiving the modified resource identifier from the reflector; and

making a request for the particular resource from the repeater designated in the modified resource identifier;

the repeater receiving the request from the client; and

returning the requested resource to the client.

26. A method as in claim **25** wherein the rewriting is performed by one of the repeater, the reflector or another repeater.

27. A method as in any one of claims **25–26** wherein the network is the Internet and wherein the resource identifier is a uniform resource locator (URL) for designating resources on the Internet, and wherein the modified resource identifier is a URL designating the repeater and indicating the reflector or origin server, and wherein the modified resource identifier is provided to the client using a REDIRECT message.

* * * * *

US006330606B1

(12) **United States Patent**
Logue et al.

(10) **Patent No.: US 6,330,606 B1**
(45) **Date of Patent: Dec. 11, 2001**

(54) **METHOD AND APPARATUS FOR DISPATCHING DOCUMENT REQUESTS IN A PROXY**

(75) Inveotors: **Jay D. Logue**, Sao Jose; **Lee S. Mighdoll**, Sao Fraocisco, both of CA (US)

(73) Assigoee: **WebTV Networks, Inc.**, Mountaio View, CA (US)

(*) Notice: Subject to aoy disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/280,920**

(22) Filed: **Mar. 29, 1999**

Related U.S. Application Data

(60) Division of application No. 08/827,643, filed on Apr. 9, 1997, now Pat. No. 5,935,207, which is a continuation-in-part of application No. 08/656,924, filed on Jun. 3, 1996, now Pat. No. 5,918,013.

(51) **Int. Cl.**[7] **G06F 15/173**

(52) **U.S. Cl.** **709/226**; 709/216; 709/218; 709/219

(58) **Field of Search** 707/10, 103; 711/118, 711/119, 122, 124; 709/213, 214, 215, 216, 218, 219, 217, 226; 725/143, 145, 148

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,325,423	6/1994	Lewis	379/90
5,488,411	1/1996	Lewis	348/8
5,538,255	7/1996	Barker	463/41
5,558,339	9/1996	Perlman	463/42
5,564,001	10/1996	Lewis	395/154
5,586,257	12/1996	Perlman	463/42
5,586,260	12/1996	Hu	395/187.01
5,612,730	3/1997	Lewis	348/8
5,675,510	10/1997	Coffey et al.	365/514 A
5,712,979	1/1998	Graber et al.	395/200.11
5,751,956	5/1998	Kirsch	395/200.33
5,754,774	5/1998	Bittinger et al.	395/200.33
5,774,670	6/1998	Montulli	395/200.57
5,787,470 *	7/1998	DeSimone et al.	711/124
5,864,852 *	1/1999	Luotonen	707/10
5,864,854 *	1/1999	Boyle	707/10
5,878,429 *	3/1999	Morris et al.	707/103
5,924,116 *	7/1999	Aggarwal et al.	711/122
5,933,849 *	8/1999	Srbljic et al.	711/118
6,112,279 *	8/2000	Wang	711/119

OTHER PUBLICATIONS

Microsoft Corporatioo, Appendix E: Web Proxy Service Reference, http://web.lnu.edu.cn/proxy/EE_map.htm, 16 pages, Jan. 1996.*

"Administrator's Guide, Netscape Proxy Server," *Netscape Communicatins Corp.* 1995, 1996, pp 19–20.

Azer Bestavros, et al., "Application–Level Document Caching in the Internet," *IEEE*, Jun., 1995, pp. 166–173.

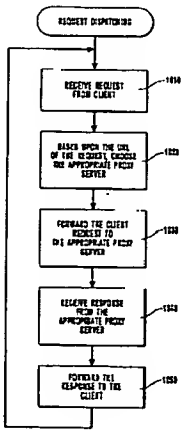
(List cootioued oo next page.)

Primary Examiner—Patrice Winder
(74) *Attorney, Agent, or Firm*—Workman, Nydegger & Seeley

(57) **ABSTRACT**

A method aod apparatus for dispatching documeot requests in a proxy to more efficiently allocate the document cache space withio the proxy is provided. A proxy includes a document cache storing recently requested documents. The proxy is coupled to a clieot aod to a remote server. A Uniform Resource–Locator–("URL")–is–included–in–the–document request. The proxy forwards the request to one of a plurality of proxy servers based upon the URL. According to aoother aspect of the preseot invention, the proxy performs a hash functioo on the URL that maps the URL to exactly one of the plurality of proxy servers. Advantageously, in this manoer, mutually exclusive portions of the Web's content can be allocated to particular proxy servers.

15 Claims, 10 Drawing Sheets-



OTHER PUBLICATIONS

Brent D. Chapman, et al., "Building Internet Firewalls, Chapter 7: Proxy Systems," O'Reilly & Associates, Inc., Sep. 1995, pp. 189–205.

Henrik Frystyk, et al., "Tutorial on Server Administration," http://www.w3.org/People/Frystyk/HttpdTutorial/, Oct., 1994, pp. 1–19.

Martin Gleeson, "pwebstats: APerl Web Statistics Generator," http://www.unimeld.edu.au/pwebstats/, Oct. 1995, pp. 1–17.

Clinton L. Jeffery, et al., "Proxy–Sharing Proxy Servers," *IEEE*, Mar., 1996, pp. 116–119.

"Caching Algorithm of CERN httpd," http.//www.w3.org/History/1994/Caching/Overview.html, Oct., 1994, pp. 1–4.

Ari Luotonen, et al., "CERN httpd 3.0 PreRelease Notes," *Cern*, http://apollo.m.ehimeu.ac.jp/ReleaseNotes_3.Opre.html, Apr., 1994, pp. 1–3.

Ari Luotonen, "Loggin Control in CERN httpd," *CERN*, http://www.unikarlsruhe.de/Betrieb/www . . . aemon/user/Config/Logging.html#AccessLog/, Dec., 1994 pp. 1–4.

Ari Loutonen, et al., "World–Wide Web Proxies," http://www.w3.org/History/1994/Proxies/Overview.html, May 24, 1994, pp. 1–20.

"Proxies", http://www.w3.org/Daemon/User/Proxies/Proxies.html, Jul., 1995, pp. 1–3.

Jeffrey Mogul, et al., "Simple Hit–Metering and Usage–Limiting for HTTP," *HTTP Working Group, Internet–Draft*, Mar. 19, 1997, pp. 1–37.

Mark L. Van Name, et al., "Proxy Servers Will Change the Web," *PC Week*, Ziff–Davis Publishing Co., full text, Mar., 1996.

* cited by examiner

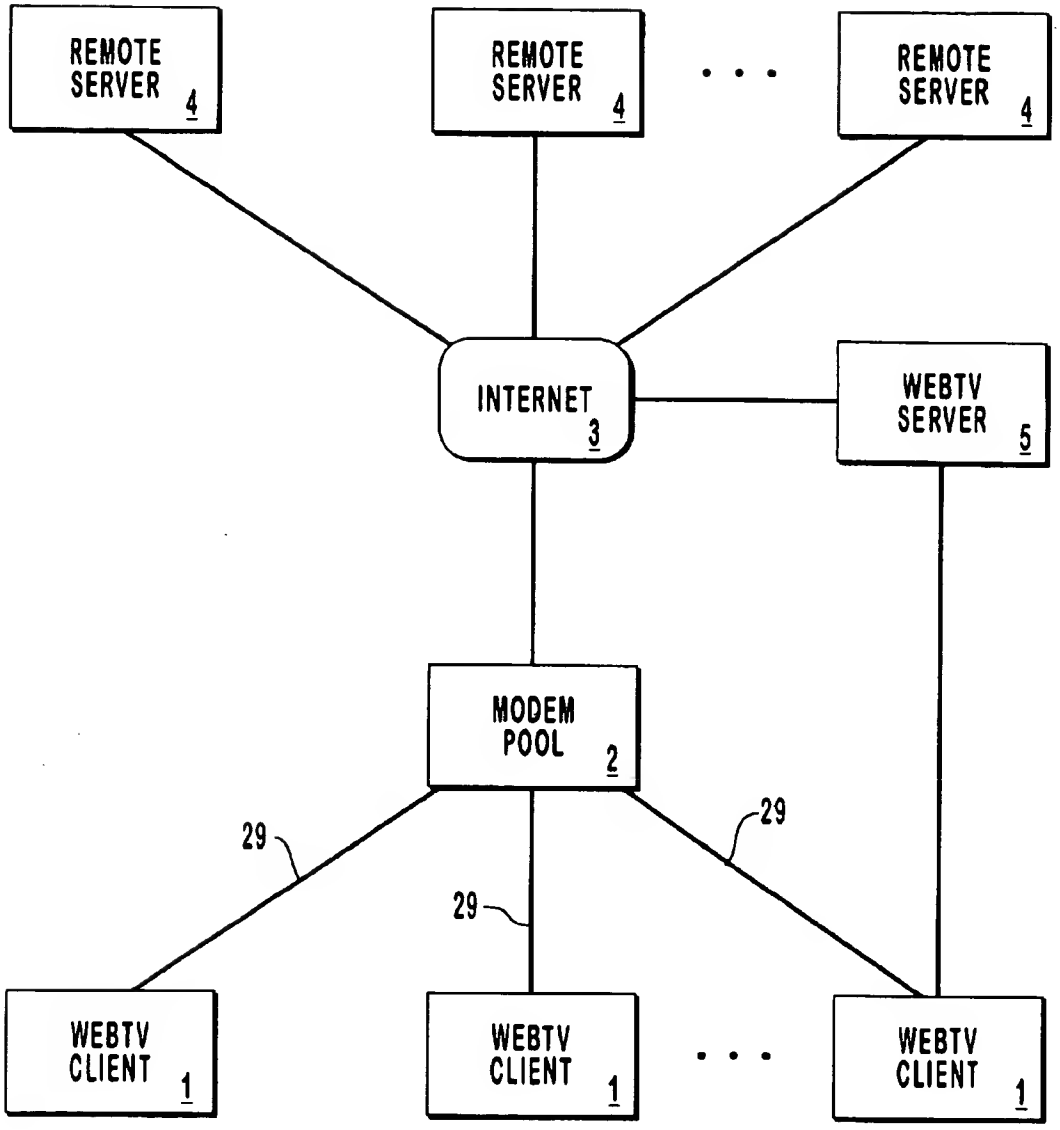

REMOTE SERVER 4
REMOTE SERVER 4
REMOTE SERVER 4
INTERNET 3
WEBTV SERVER 5
MODEM POOL 2
29
29
29
WEBTV CLIENT 1
WEBTV CLIENT 1
WEBTV CLIENT 1

FIG. 1

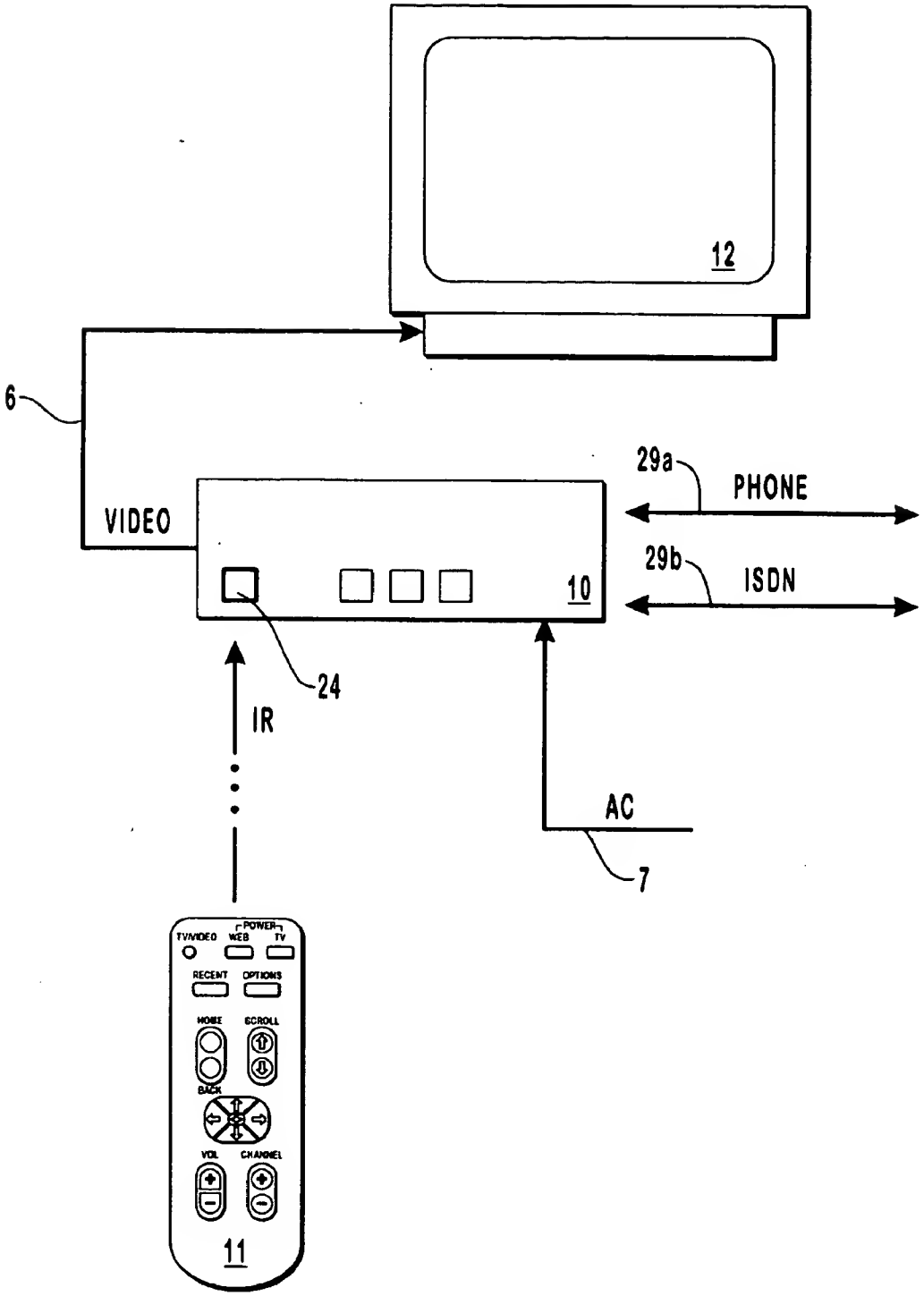


FIG. 2

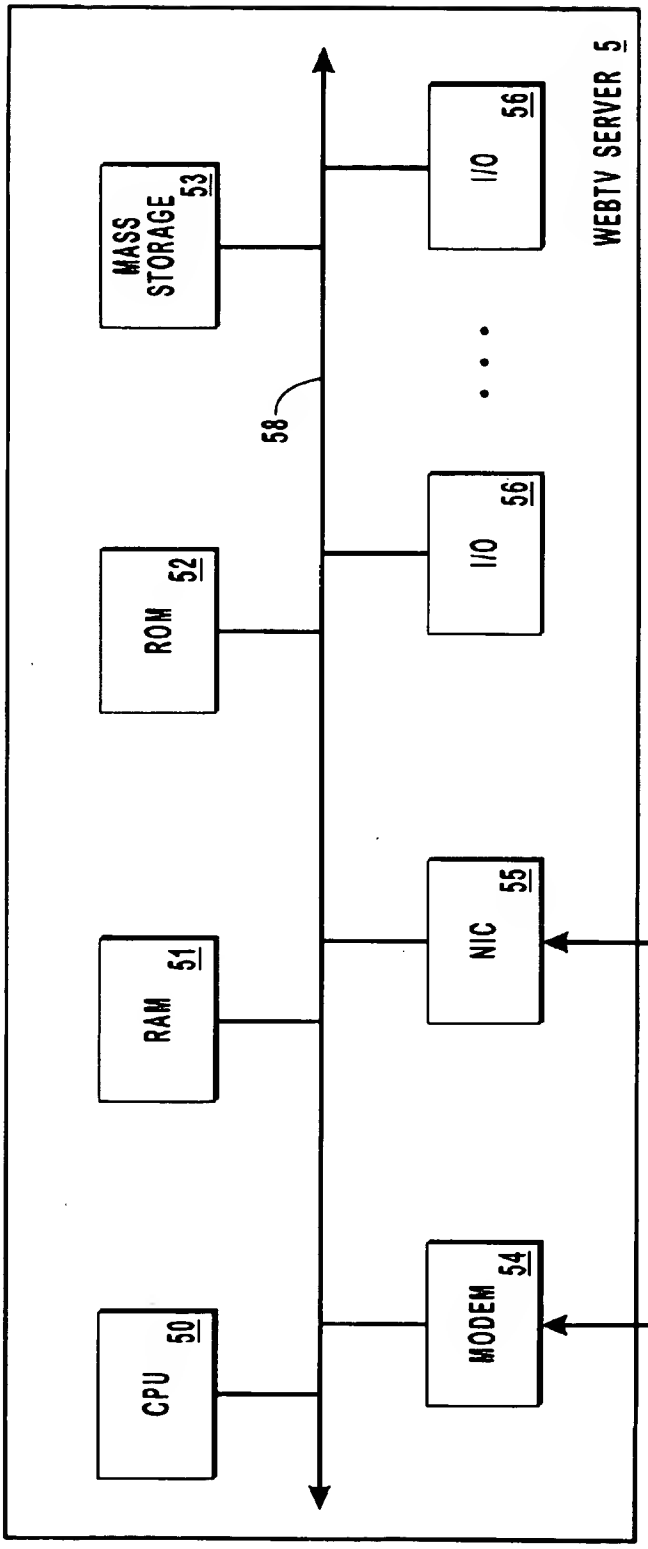

WEBTV SERVER 5
MASS STORAGE 53
I/O 56
58
ROM 52
I/O 56
RAM 51
NIC 55
TO OTHER WEBTV SERVICES
CPU 50
MODEM 54
TO INTERNET

FIG. 3

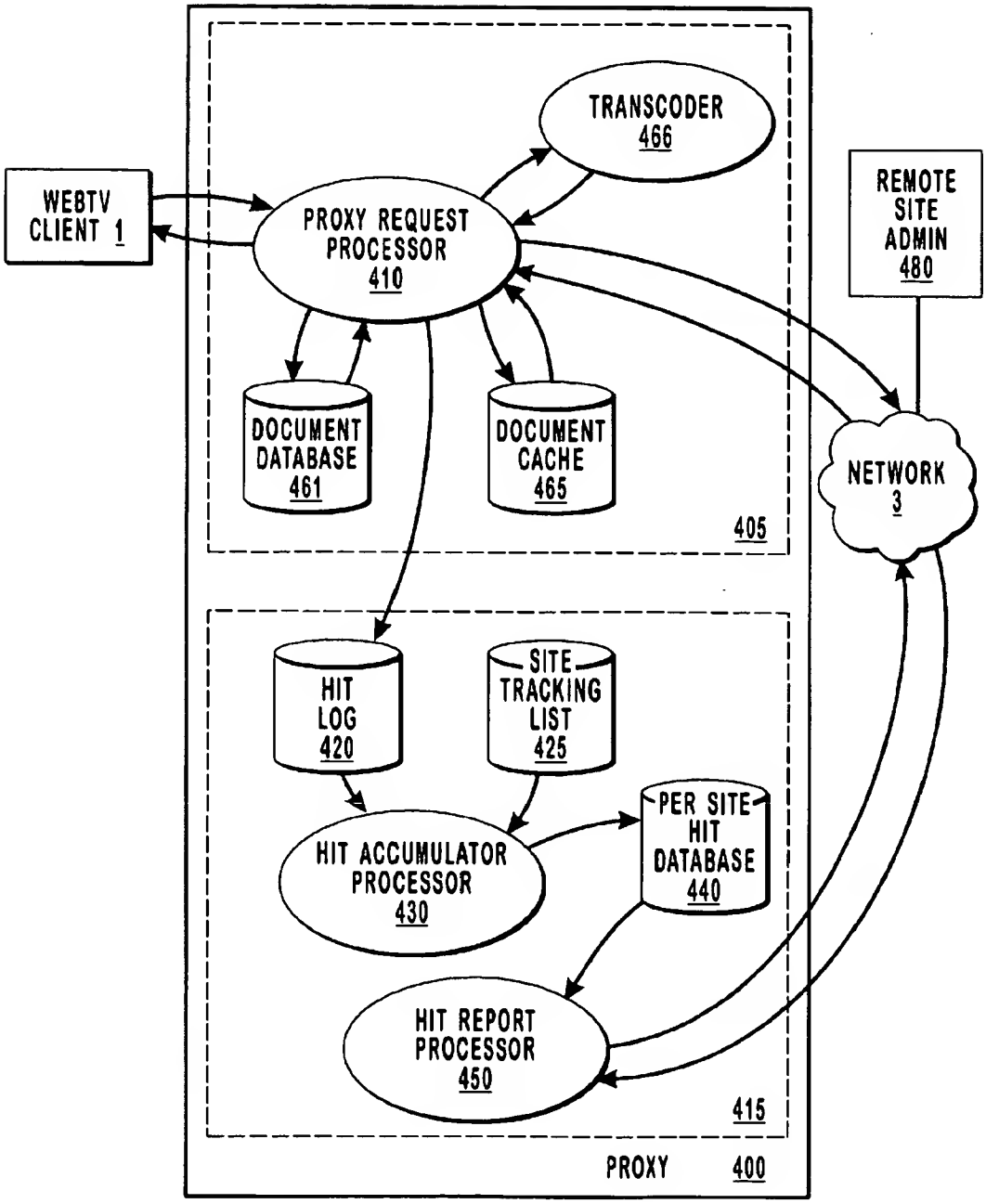

TRANSCODER
466
WEBTV
CLIENT 1
PROXY REQUEST
PROCESSOR
410
REMOTE
SITE
ADMIN
480
DOCUMENT
DATABASE
461
DOCUMENT
CACHE
465
NETWORK
3
405
HIT
LOG
420
SITE
TRACKING
LIST
425
PER SITE
HIT
DATABASE
440
HIT ACCUMULATOR
PROCESSOR
430
HIT REPORT
PROCESSOR
450
415
PROXY 400

FIG. 4

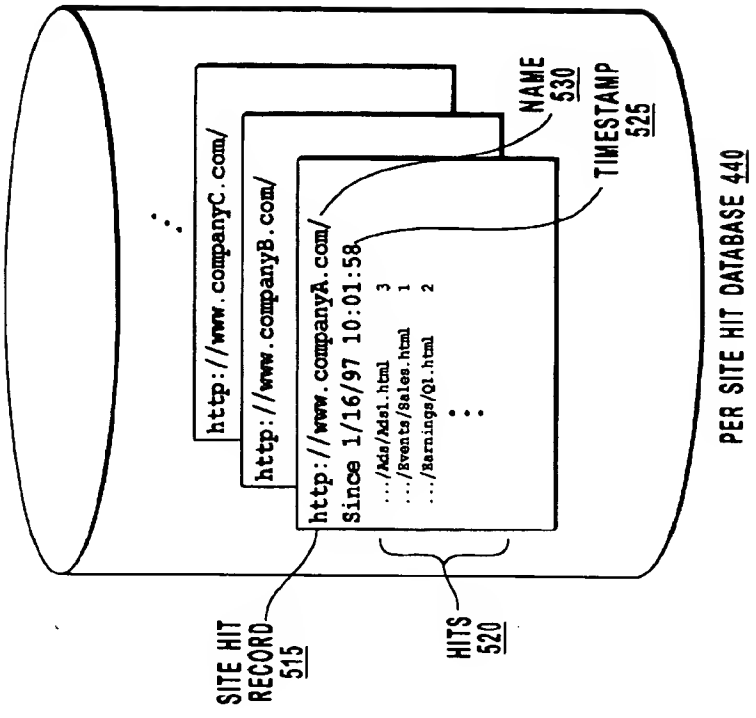

http://www.companyC.com/
http://www.companyB.com/
http://www.companyA.com/
Since 1/16/97 10:01:58
.../Ads/Ad1.html 3
.../Events/Sales.html 1
.../Earnings/Q1.html 2
NAME 530
TIMESTAMP 525
PER SITE HIT DATABASE 440
SITE HIT RECORD 515
HITS 520

FIG. 5B

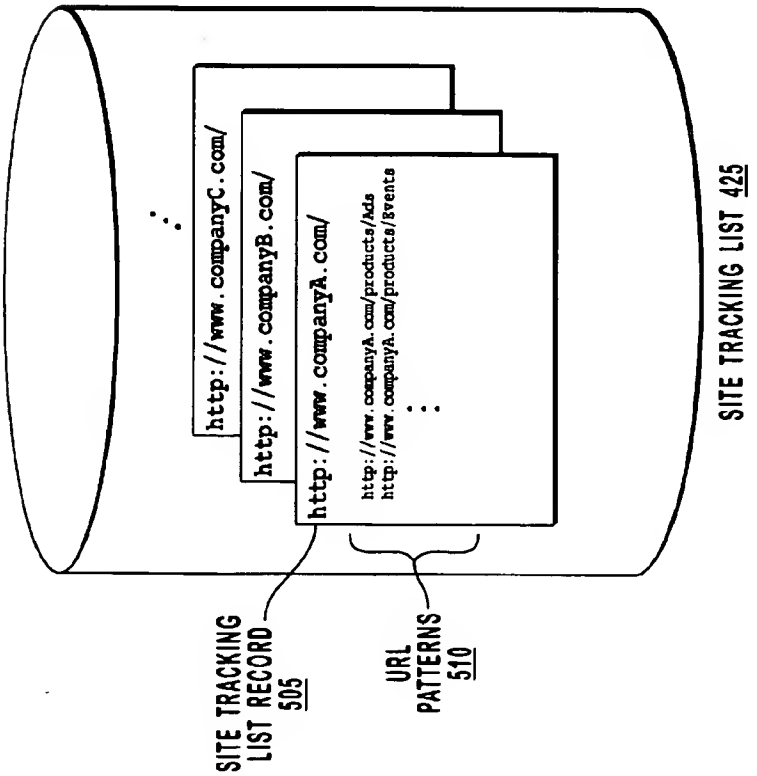

http://www.companyC.com/
http://www.companyB.com/
http://www.companyA.com/
http://www.companyA.com/products/Ads
http://www.companyA.com/products/Events
SITE TRACKING LIST 425
SITE TRACKING LIST RECORD 505
URL PATTERNS 510

FIG. 5A

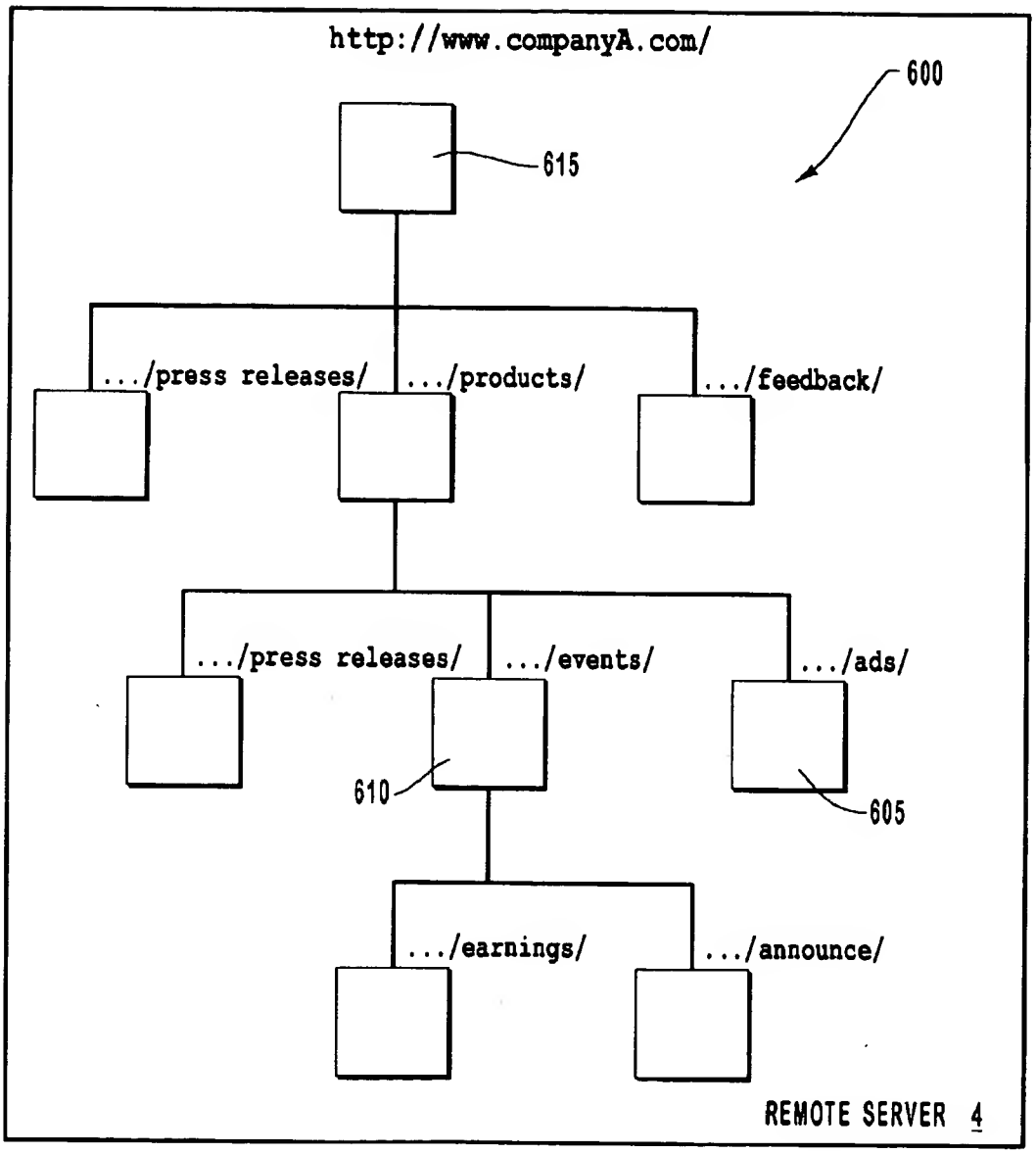

http://www.companyA.com/
600
615
.../press releases/
.../products/
.../feedback/
.../press releases/
.../events/
.../ads/
610
605
.../earnings/
.../announce/
REMOTE SERVER 4

FIG. 6

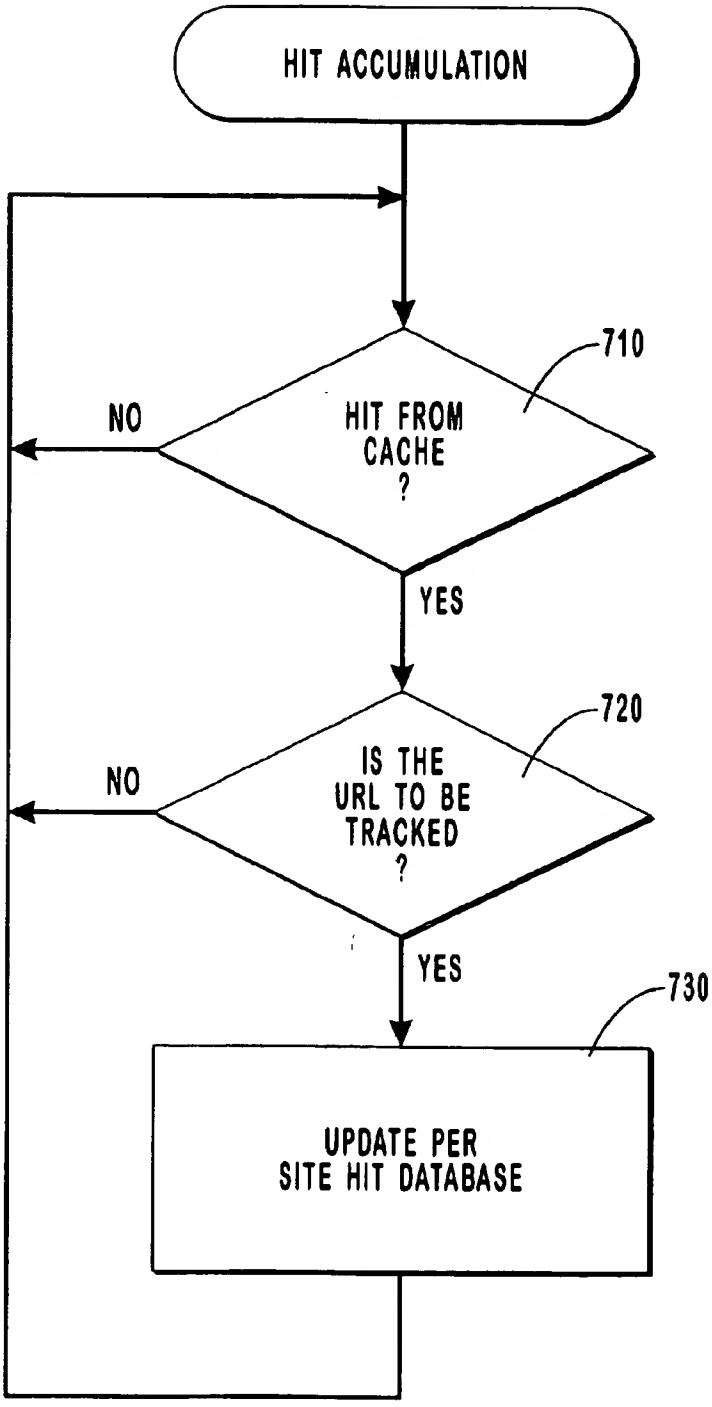

HIT ACCUMULATION
710
HIT FROM CACHE ?
NO
YES
720
IS THE URL TO BE TRACKED ?
NO
YES
730
UPDATE PER SITE HIT DATABASE

FIG. 7

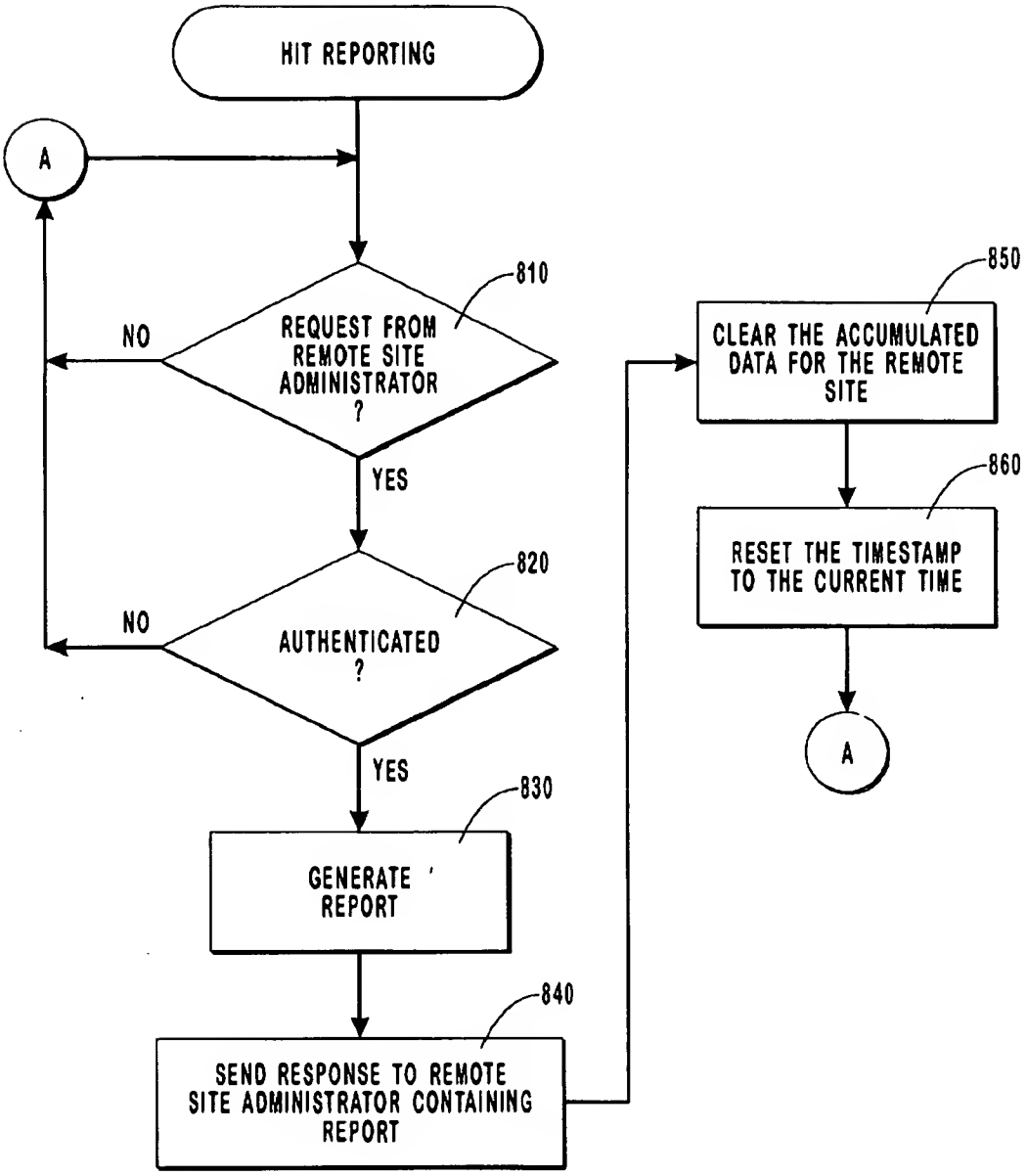

HIT REPORTING
A
810
REQUEST FROM REMOTE SITE ADMINISTRATOR ?
NO
YES
820
AUTHENTICATED ?
NO
YES
830
GENERATE REPORT
840
SEND RESPONSE TO REMOTE SITE ADMINISTRATOR CONTAINING REPORT
850
CLEAR THE ACCUMULATED DATA FOR THE REMOTE SITE
860
RESET THE TIMESTAMP TO THE CURRENT TIME
A

FIG. 8

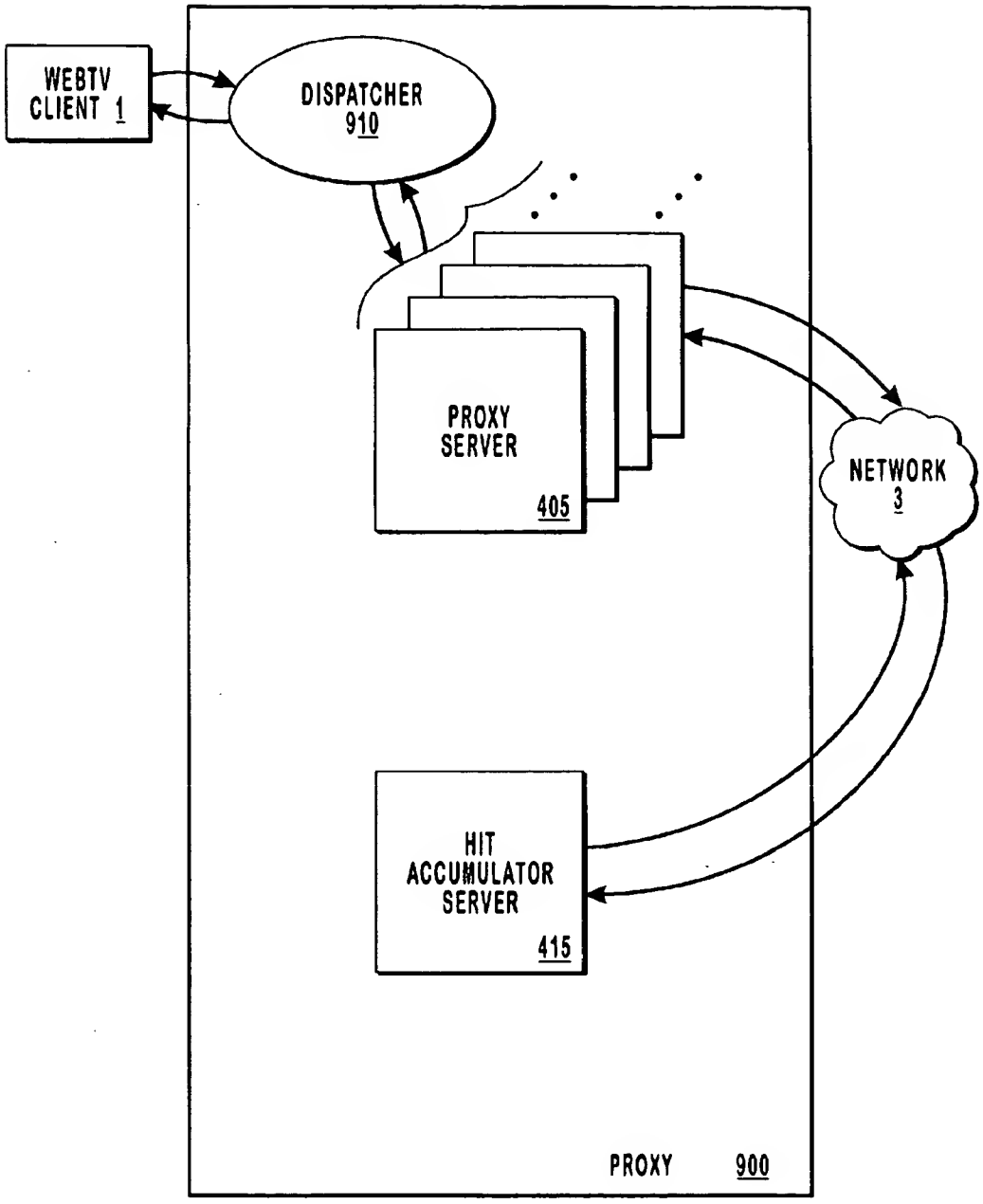

WEBTV CLIENT 1
DISPATCHER 910
PROXY SERVER 405
NETWORK 3
HIT ACCUMULATOR SERVER 415
PROXY 900

FIG. 9

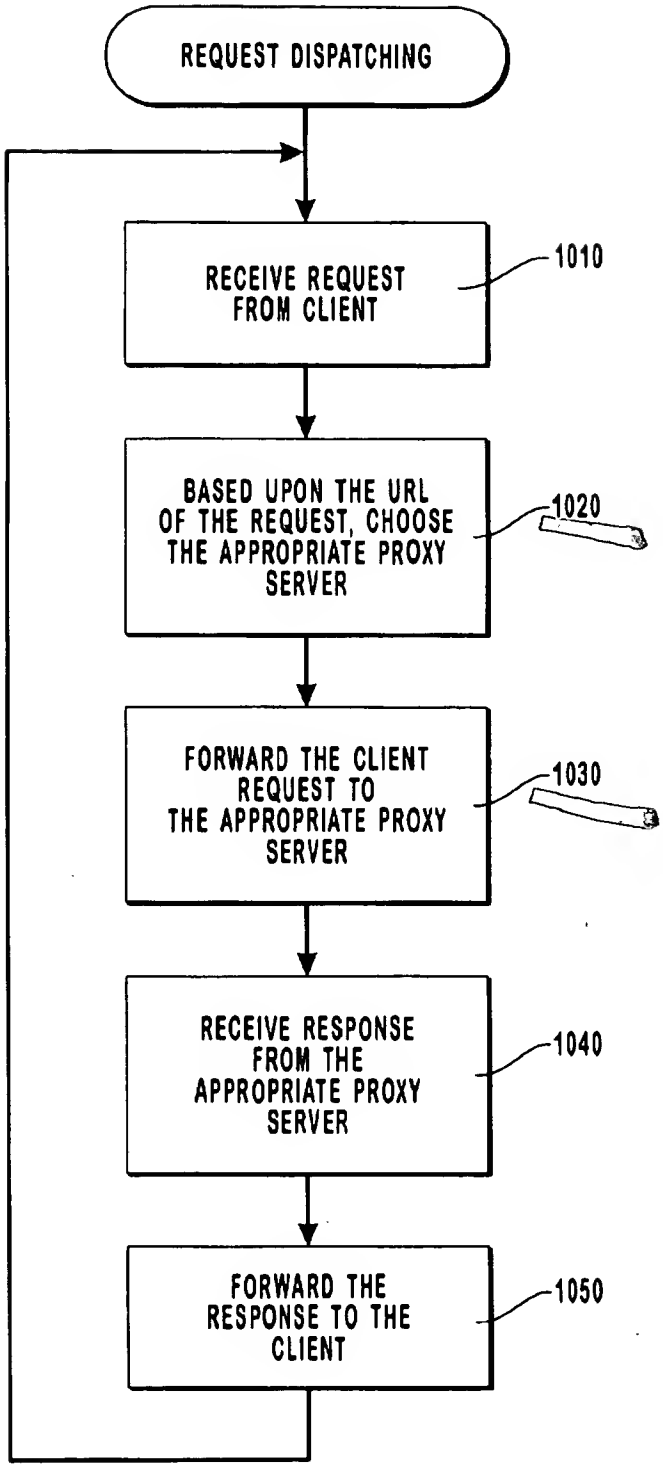

REQUEST DISPATCHING
RECEIVE REQUEST FROM CLIENT
1010
BASED UPON THE URL OF THE REQUEST, CHOOSE THE APPROPRIATE PROXY SERVER
1020
FORWARD THE CLIENT REQUEST TO THE APPROPRIATE PROXY SERVER
1030
RECEIVE RESPONSE FROM THE APPROPRIATE PROXY SERVER
1040
FORWARD THE RESPONSE TO THE CLIENT
1050

FIG. 10

METHOD AND APPARATUS FOR DISPATCHING DOCUMENT REQUESTS IN A PROXY

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a divisional of U.S. patent application entitled "Method and Apparatus for Providing Remote Site Administrators with User Hits on Mirrored Web Sites," having application Ser. No. 08/827,643, and filed on Apr. 9, 1997, now U.S. Pat. No. 5,935,207 which is a continuation-in-part of U.S. patent application entitled, "Method and Apparatus for Providing Proxying and Transcoding of Documents in a Distributed Network," having application Ser. No. 08/656,924, and filed on Jun. 3, 1996 now U.S. Pat. No. 5,918,013. The foregoing patents and patent applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates generally to the field of client-server computer networking. More specifically, the invention relates to a method and apparatus for dispatching document requests in a proxy.

2. The Prior State of the Art

World Wide Web (Web) documents are commonly written in Hypertext Mark-up Language (HTML). HTML documents typically reside on Web servers and are requested by Web clients. Often, delays can be introduced during Web browsing, for example, by heavy communications traffic on the Internet or by a slow response of a remote site. Providing one or more servers for mirroring Web sites located on remote servers is one means of reducing delays involved with browsing the Web. These mirroring servers, typically referred to collectively as a "proxy" or individually as "proxy servers," store frequently accessed Web sites in a local cache, thereby eliminating recurrent retrievals of commonly accessed documents. Thus, when a request for a particular Web page is received from a client, the proxy server associated with the particular client looks first to its local cache to service the request rather than the remote site upon which the Web page resides. If the requested document is found locally, the request can be serviced by the proxy server and a subsequent request to the remote server for the document can be avoided. Therefore, only when a valid copy of the requested document is not in the proxy's local cache would the remote server need to be accessed. In this manner, exposure to heavy communications traffic on the Internet and slow responses of remote serves can be reduced.

While this mirroring approach is beneficial to end-users, the proxy's cache space is inefficiently allocated in current mirroring technology. Currently, each client is assigned to one or more proxy servers. Therefore, the documents most recently requested by each active client will reside in the corresponding proxy server's cache. Assuming one or more clients assigned to different proxy servers have requested the same document recently, the same document might be cached in several of the proxy servers, thereby reducing the cache storage space for other frequently requested documents. Further, one or more extremely popular documents might potentially be cached in each proxy server. While redundancy of information is useful for fault tolerance, organized redundancy would be preferable. Given the foregoing, what is needed is a means of more efficiently allocating cache space within a proxy. Specifically, it would be desirable to allocate mutually exclusive portions of the Web's content to particular proxy servers.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described for dispatching document requests in a proxy to more efficiently allocate the document cache space within the proxy. A proxy includes a document cache storing recently requested documents. The proxy is coupled to a client and to a remote server. The proxy implements a dispatching scheme for client requests that results in a more efficient allocation of the proxy's document cache space. The proxy receives a document request from a client. A Uniform Resource Locator (URL) is included in the document request. The proxy forwards the request to one of a plurality of proxy servers based upon the URL.

According to another aspect of the present invention, the proxy performs a hash function on the URL that maps the URL to exactly one of the plurality of proxy servers. Advantageously, in this manner, mutually exclusive portions of the Web's content can be allocated to particular proxy servers.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. **1** is a block diagram illustrating several clients connected to a proxy server in a network.

FIG. **2** is a diagram illustrating a client according to one embodiment of the present invention.

FIG. **3** is a block diagram of a server according to one embodiment of the present invention.

FIG. **4** is a data flow diagram illustrating the interaction of proxy components according to one embodiment of the present invention.

FIG. **5A** is a depiction of an exemplary site tracking list according to one embodiment of the present invention.

FIG. **5B** is a depiction of an exemplary per site hit database according to one embodiment of the present invention.

FIG. **6** is a logical view of an exemplary directory structure of a remote server.

FIG. **7** is a flow diagram illustrating a method of performing hit accumulation according to one embodiment of the present invention.

FIG. **8** is a flow diagram illustrating a method of hit reporting according to one embodiment of the present invention.

FIG. **9** is a data flow diagram illustrating the interaction of proxy components according to another embodiment of the present invention.

FIG. **10** is a flow diagram illustrating a method of dispatching requests to segregate the storage of documents according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for maintaining a more efficient document caching scheme in a client-server computer network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these

specific details. Further, in other instances, well-known structures and devices are shown in hlock diagram.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

While emhodiments of the present invention will be described with respect to HTML documents, the method and apparatus described herein are equally applicable to other types of documents such as text files, images (e.g., JPEG and GIF), audio files (e.g., .WAV, AU, and AIFF), video files (e.g., .MOV, and AVI), and other document types commonly found on the Web.

Sytem Overview

The present invention may he included in a system, known as WehTV™, for providing a user with access to the Internet. A user of a WehTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Weh, send and receive electronic mail (e-mail), and use various other WehTV™ network services. The WehTV™ network services are provided by WehTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. **1** illustrates a hasic configuration of the WehTV™ network according to one emhodiment. A numher of WebTV™ clients **1** are coupled to a modem pool **2** via direct-dial, hi-directional data connections **29**, which may he telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool **2** is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers **4** via a conventional network infrastructure **3**, such as the Internet. The WebTV™ system also includes a WebTV™ server **5**, which specifically supports the WebTV™ clients **1**. The WebTV™ clients **1** each have a connection to the WebTV™ server **5** either directly or through the modem pool **2** and the Internet **3**. Note that the modem pool **2** is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation, the WehTV™ server **5** is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as heing performed hy such single device. However, the WehTV™ server **5** may actually comprise multiple physical and logical devices connected in a distrihuted architecture, and the various functions discussed below which are provided by the WehTV™ services may actually he distrihuted among multiple WehTV™ server devices.

An Exemplary Client System

FIG. **2** illustrates a WehTV™ client **1**. The WebTV™ client **1** includes an electronics unit **10** (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set **12**, and a remote control **11**. In an alternative emhodiment of the present invention, the WehTV™ hox **10** is built into the television set **12** as an integral unit. The WehTV™ box **10** includes hardware and software for providing the user with a graphical user interface, hy which the user can access the WebTV™ network services, hrowse the Weh, send e-mail, and otherwise access the Internet. The WehTV™ client **1** uses the television set **12** as a display device. The WebTV™ hox **10** is coupled to the television set **12** by a video link **6**. The video link **6** is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client **1** includes both a standard modem and an ISDN modem, such that the communication link **29** between the WehTV™ box **10** and the server **5** can he either a telephone (POTS) connection **29**a or an ISDN connection **29**b. The WehTV™ box **10** receives power through a power line **7**.

Remote control **11** is operated by the user in order to control the WebTV™ client **1** in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box **10** receives commands from remote control **11** via an infrared (IR) communication link. In alternative embodiments, the link between the remote control **11** and the WebTV™ box **10** may be RF or any equivalent mode of transmission.

An Exemplary Server System

The WebTV™ server **5** generally includes one or more computer systems generally having the architecture illustrated in FIG. **3**. It should he noted that the illustrated architecture is only exemplary; the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) **50**, random access memory (RAM) **51**, read-only memory (ROM) **52**, a mass storage device **53**, a modem **54**, a network interface card (NIC) **55**, and various other input/output (I/O) devices **56**. Mass storage device **53** includes a magnetic, optical, or other equivalent storage medium. I/O devices **56** may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem **54** is used to communicate data to and from remote servers **4** via the Internet.

As noted above, the WebTV™ server **5** may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC **55** is used to provide data communication with other devices that are part of the WehTV™ services. Modem **54** may also be used to communicate with other devices that are part of the WehTV™ services and which are not located in close geographic proximity to the illustrated device.

An Exemplary Proxy

FIG. **4** illustrates the caching and hit accumulation features of the WebTV™ proxy **400** according to one embodiment of the present invention. In this embodiment, one or more WebTV™ servers **5** may act as a proxy **400** in providing the WebTV™ client **1** with access to the Web and other WebTV™ services. More specifically, WebTV™ server **5** functions as a "caching proxy." In this example, proxy **400** includes a proxy server **405** and a hit accumulator server **415**. Client requests that are serviced from the proxy server's local document cache **465** are communicated to the hit accumulator server **415**. As will be described below, the hit accumulator server **415** maintains and organizes the data so as to provide hit tracking information to remnte site administrators such as remote site administrator **480**. Remote site administrator **480** may include entities such as persons authorized to gather statistical data for the remote

site, persons authorized to manage and maintain the remote site, the remote site itself, or an automated computer system or other device configured to receive statistical data for the remote site.

In this embodiment, the proxy server **405** includes a proxy request processor **410**, a document cache **465**, a document database **461**, and a transcoder **466**. The proxy request processor **410** receives requests from the WebTV™ client **1** and sends responses to the WebTV™ client **1**. The proxy request processor **410** maintains the document database **461**, the document cache **465**, and further determines when transcoding will be performed. The document cache **465** is used for temporary storage of Web documents such as images, text files; audio files, video files and other information which is used frequently by either WebTV™ client **1** or the proxy server **405**.

When a document request is received from a client, the proxy request processor **410** determines whether to service the request from the document cache **465** by performing a search of the document cache **465**. If the document is found locally, then the document may be retrieved from the document cache **465** and transferred to the client with the response. However, if the requested document is not found, then the proxy request processor **410** requests the document from the appropriate site and upon receipt the proxy request processor **410** provides the document to the client with the response. Further, the proxy request processor **410** anticipates subsequent requests by storing the document in the document cache **465**.

When a document is retrieved by the proxy server **405** from a remote server **4**, for example, detailed information on this document may be stored in the document database **461**. The stored information may subsequently be used by the proxy server **405** to speed up processing and downloading of that document in response to future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service may be facilitated by making use of information stored in the document database **461**. For example, the document database **461** may include certain historical and diagnostic information for Web pages that have been accessed by a WebTV™ client **1**.

Document transcoder **466** is used to automatically revise the code of Web documents retrieved from the remote servers **4**, for purposes such as: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client **1**; (3) improving the efficiency of transmission of documents from the server **5** to the client **1**; (4) matching hardware decompression technology within the client **1**; (5) resizing images to fit on the television set **12**; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client **1** when displaying a Web page with in-line images (images displayed in text); and (8) altering documents to fit into smaller memory spaces.

In one embodiment, hit accumulator server **415** may act as a Web server providing a Hypertext Transport Protocol (HTTP) interface by which remote site administrators can access the accumulated hits for their sites by way of a Web browser. The hit accumulator server **415** may include a hit log **420**, a hit accumulator processor **430**, a site tracking list **425**, a hit report processor **450**, and a per site hit database **440**. One method of communicating hits from a given proxy server to the hit accumulator server **415** is through a common storage device such as hit log **420**. This and other methods of communicating hits will be described below. Regardless of how hits are communicated to the hit accumulator server **415**, a processor such as the hit accumulator processor **430** is desirable to verify the hits against a list of locations that are to be monitored. Such a list of locations may be stored in the site tracking list **425**, for example. A location, in this context, refers to the location of a document. The location may be represented by a URL, a directory path, or other mechanisms for uniquely identifying a particular document. Hits that are validated by the hit accumulator processor **430** are recorded in the per site hit database **440**. Thus, the per site bit database **440** will have a current count of the bits for each location listed in the site tracking list **425**. In this embodiment, the hit report processor **450** may receive requests from remote site administrators such as remote site administrator **480** for bit reports. The hit reports can be extracted from the per site hit database **440** and transmitted to the requester in an HTML report, for example.

While in this embodiment the proxy server **405** and the hit accumulation server **415** have been shown as separate servers, the functionality of both could be combined into one WebTV™ server **5**. Additionally, the proxy **400** might be expanded to include more than one proxy server **405**. When expanding the proxy **400** to include more than one proxy server **405**, only one hit accumulation server **415** need be employed.

In alternative embodiments, hits may be communicated by a proxy server **405** to the accumulation server **415** by way of a network connection such as permanent connection through which events may be sent. Also, message passing may be employed whereby the proxy server **405** sends a message such as a datagram to the hit accumulator **415** to notify it of a document cache hit. It is appreciated that many other means of communicating information between servers are possible.

An Exemplary Site Tracking List

FIG. **5A** illustrates an exemplary site tracking list according to one embodiment of the present invention. This illustration depicts a site tracking list **435** including site tracking list records **505** for three remote sites: (1) http://www.companyA.com/; (2) http://www.companyB.com/; and (3) http://www.companyC.com/. In this embodiment, each site tracking list record **505** may include a list of one or more URL patterns **510**.

The list of URL patterns **510** may be a list of strings identifying the initial portions (e.g., prefixes) of URLs to be tracked. In this example, the proxy **400** tracks hits for documents identified by URLs with a prefix that matches any of the URL patterns **510** specified in one of the site tracking list records **505**. The hits may then be logged to a record in the per site hit database **440** corresponding to the site tracking list record **505** which contained the matching URL pattern. This form of URL pattern is useful for tracking hits for a particular grouping of Web pages beginning with the same initial sequences of characters. For example, the URLs for the Web pages of Company A might all begin with "http://www.companyA.com/." Additionally, the Web pages associated with products produced by Company A might all begin with the sequence "http://www.companyA.com/product/." Furthermore, pages related to a particular product might all begin with the URL prefix "http://www.companyA.com/product/<product_name>/" where <product_name> identifies the particular product. To track the hits for pages relating to Company A's Gizmo product line, therefore, the following URL pattern may be used: "http://www.companyA.com/product/Gizmo/." Similarly, to track the hits for all of Company A's products the following URL pattern may be used: "http://www.companyA.com/product/."

URL patterns are not limited to prefixes, other forms of URL patterns may be used such as patterns including wild card or other special characters, or patterns in the form of standard regular expressions.

An Exemplary per Site Hit Database

FIG. 5B illustrates an exemplary per site hit database according to one embodiment of the present invention. Based upon the information provided in the site tracking list **425** of FIG. 5A, an exemplary per site hit database might be represented as per site hit database **440**. In this example, the per site hit database **440** includes three site hit records **515** corresponding to remote sites for CompanyA, CompanyB and CompanyC.

In this embodiment, each site hit record **515** includes a timestamp **525**. The a timestamp **525** may indicate the time from which the hits have been accumulated. In this example, therefore, there have been six hits to the monitored URLs since Jan. 16, 1997 at 10:01:58. Those of skill in the art will appreciate the timestamp **525** may represent the period of accumulation in other ways such as elapsed time since the last hit report was generated.

Site hit records **515** also include a remote site name **530**. The remote site names **530** from front to back correspond to CompanyA, CompanyB, and CompanyC. Site hit record **515** further includes a list of hits **520**. In this embodiment, the list of hits **520** includes the URLs of the documents that were requested and subsequently serviced from the proxy's local cache (e.g., document cache **465**) since the time indicated hy the timestamp **525**. According to the site hit record **515** for CompanyA, the adl.html Web page has been requested and serviced from the proxy's local cache three times. Similarly, the sales.html and Ql.html Web pages have been provided from the proxy's cache once and twice, respectively. Based upon the accumulated hit information in a particular site hit record **515**, a detailed hit report may be provided to the corresponding remote site administrator. Hit accumulation will he discussed further helow.

FIG. **6** is a logical view of an exemplary directory structure **600** that may exist on a remote server **4**. This exemplary directory structure **600** illustrates the need for a flexihle method of tracking the numher of hits. Web pages might reside in any or all of the directories shown. In this example, the URL patterns within a site tracking list record **505** may identify a particular directory or directories in the hierarchy depicted.

The remote site administrator for CompanyA may want to know the number of hits in an Ads suhdirectory **605** and an Events suhdirectory **610**. This may he due to the fact that advertising hanners are shown on Web pages in these directories and the advertisers may want feedhack on how many Web viewers are seeing their ads. Alternatively, the company may have its own husiness reasons for analyzing statistics in certain areas of their Web site. Regardless, it is apparent that simply tracking all hits for a root directory **615** on the company's server is insufficient. For example, hits would be tracked for directories in which the remote site administrator had no interest. A list of URL patterns is used to accommodate the flexibility desired. The following URL patterns may be stored in the site tracking list **425** for CompanyA to track the above-mentioned suhdirectories: "http://www.companyA.com/products/Events/" and "http://www.companyA.com/products/Ads/." The list of URL patterns **510** in each site tracking list record **505** allows a remote site to enumerate specific directories, for example, in which they would like to track user hits.

The advantages of providing forms of URL patterns with wild cards becomes apparent with reference to the directory structure **600**. Assume the "*" character is a wild card. That is, it matches zero or more characters. Since, CompanyA has two subdirectories with press releases, a convenient way to track hits in both is with the following URL pattern: "http://www.companyA.com/*press_releasesA/." Without the use of a wild card, the equivalent URL patterns are as follows: "http://www.companyA.com/press_releases/" and "http://www.companyA.com/products/press_releases." Thus, it should be appreciated that wild cards and regular expressions provide additional efficiency and convenience in the specification of URL patterns.

Hit Accumulation

FIG. **7** is a flow diagram illustrating a method of performing hit accumulation according to one embodiment of the present invention. In this embodiment, each site hit record **515** begins in an initial state having an indication of the remote site (e.g., the name **530**) and a timestamp **525** representing the time at which hit accumulation hegan. Initially, the hit accumulation server **415** waits for an indication that a client request has heen serviced from the proxy's local cache (step **710**). For example, the hit accumulator processor **430** may determine that a new entry has heen made to the hit log **420**.

Upon receiving an indication that the proxy **400** has served up a cached response, the hit accumulation server **415** determines if the URL of the document retrieved from the proxy's local cache is one whose hits are to be tracked. As discussed above, not all hits are tracked. In this embodiment, hits are tracked only for documents matching URL patterns that have heen registered in a tracking list such as the site tracking list **425**, discussed above. Therefore, the hit accumulator processor **430** compares the URL of the retrieved document to URL patterns **510** in each site tracking list record **505** to determine if the hit will be recorded in the per site hit database **440** (step **720**). If no URL patterns **510** match the retrieved document the hit is ignored. Otherwise, if the retrieved document matches any of the URL patterns **510**, then the appropriate site hit record **515** in the per site hit database **440** is updated (step **730**).

Update of the site hit record **515** can he explained briefly with respect to FIG. 5B In this embodiment, the appropriate site hit record **515** is searched for an entry that matches the URL of the retrieved document. If the retrieved document's URL does not already exist in the list of hits **520** for the site hit record **515**, then the URL is added and its count is set to one since this is the document's first hit. However, if the retrieved document's URL was already in the list of hits **520** (meaning it has had at least one previous hit), then only the corresponding count needs to be incremented. In this manner, each document retrieved from the proxy's local cache that matches a tracked URL pattern will have an entry in the list of hits **520** with a corresponding count indicating the number of cache hits.

Hit Reporting

Referring now to FIG. **8**, a method of hit reporting according to one embodiment of the present invention is illustrated. In this embodiment of the present invention, the hit accumulator server **415**, in addition to its other responsihilities, acts as a Web server providing an HTTP interface by which remote site administrators can access the accumulated hits for their respective tracked sites. The hit report processor **450** waits until a request is received from a

remote site administrator (step **810**). Preferably, the HTTP address on the hit accumulation server **415** can be used to identify the requester of the information. For example, the hit report for Company A, might be accessed on the hit accumulation server **415** at: "http://www.webtv.net/hits/company_a."

To limit access to the hit reports a secure communication technology such as Secure Sockets Layer (SSL) or other available secure communication protocol can be used to keep the hit information private by providing encrypted communications across the network. Additionally, the report requests can be authenticated to assure only a particular remote server or individual can access the information (step **820**).

Once a request has been received from a remote site administrator and it has been optionally authenticated, then a report can be generated from the hit data accumulated such as the list of hits **520** for the particular site hit record **515** (step **830**). In this embodiment, the report may include a list of URLs and their corresponding counts since the last report.

For convenient access via the Web, the report may be formatted in an HTML format. Also, for the convenience of the remote site administrator, a timestamp that identifies the starting point of the accumulation may be included in the report. The level of specificity of the URL list may be at the document level thereby allowing the remote site administrator to determine the number of hits for individual documents. However, it may also be helpful to additionally summarize the hits by directory, for example. It will be recognized that numerous other ways of formatting and arranging the hit reports are possible.

After the report has been formatted, the response containing the report is transmitted to the remote site administrator (step **840**).

In this embodiment, before resuming the hit accumulation of FIG. **7**, the accumulated data in the site hit record **515** is cleared (step **850**) also the timestamp **525** is reset to reflect the current time. The above steps for retrieving a report from the proxy may be periodically repeated at the convenience of the remote site administrator whenever an accurate total hit count is desired.

In alternative embodiments, hit reports may be provided to remote sites in a number of other ways. Hit reports need not be initiated by a request from the remote site administrator. For example, the proxy may periodically send unsolicited hit reports via email, the proxy may periodically download hit updates to a device specified by the remote site administrator, or the hit reports might be transmitted to remote site administrators in the form of datagrams. In any event, the assignees of the present invention appreciate a variety of reporting mechanisms are possible.

Allocation of Cache Space within a Proxy

FIG. **9** is a data flow diagram illustrating the interaction of proxy components according to another embodiment of the present invention. In this embodiment, proxy **900** includes a plurality of proxy servers **405** communicatively coupled to a dispatcher **910** and a hit accumulator server **415**. Rather than allowing a given proxy server's cached contents to be determined based upon the requests of an associated client, the content of the Web can be distributed among proxy servers **405** by a hash algorithm executed by the dispatcher **910**. The hash algorithm preferably maps a given URL to one and only one of the plurality of proxy servers **405**. This can be accomplished using a portion of the output of a secure hash algorithm such as the Message Digest 5 (MD5) hash algorithm. The hash algorithm can be thought of as a mechanism for assigning a range of URLs to each of the proxy servers **405** in the proxy **900**.

In this embodiment, the dispatcher **910** receives document requests including URLs from a client such as WebTV™ client **1**. Based upon the URL in the request, the dispatcher **910** determines the proxy server **405** in which the document should be cached and forwards the client request to that proxy server **405**. If the document requested by the client is not found in the proxy server's local document cache **465**, then the proxy server **405** requests the document from the appropriate server (e.g., a remote server) and caches the document when it is received from the server.

If redundancy is desired, the hashed result of a URL may be used to identify a cluster of two or more proxy servers rather than a single proxy server **405**. In this manner, the load required to support a popular document can be shared among a group of proxy servers.

In an alternative embodiment, a decentralized dispatching scheme can be implemented. For example, the proxy servers **405** may be arranged to form an interconnected ring configuration and the functionality of the dispatcher **910** may be incorporated into each proxy server **405**. In this embodiment, the client document requests may be initially handled by one of the proxy servers **405** in the ring. If the requested document is not found in the local cache of the initial proxy server, the initial proxy server may forward the request to the appropriate proxy server based on the hashing scheme discussed above.

FIG. **10** is a flow diagram illustrating a method of dispatching requests to segregate the storage of documents according to one embodiment of the present invention. While both a centralized and a decentralized request dispatching mechanism have been discussed above, the method described below is generally applicable to both. In this embodiment, initially, a document request is received from a client (step **1010**). If a centralized dispatcher such as dispatcher **910** receives the request, then based upon the URL an appropriate proxy server is determined based upon the output of the hash algorithm (step **1020**).

However, in a decentralized dispatching environment, the initial proxy server receiving the client request may assume it is the appropriate proxy server and first check its local document cache **465**. If the document is not present, then proxy server may perform the hash algorithm on the URL to determine which of the remaining proxy servers is appropriate for the request (step **1020**).

After determining the proxy server appropriate for the client request, the request is forwarded to that proxy server (step **1030**). The proxy server **405** attempts to service the request from its local document cache **465**. If a cache hit occurs, then the document is immediately available from the proxy server's local document cache **465**. However, if a cache miss occurs, the proxy server **405** will retrieve the document from an appropriate server and store a copy locally. In any event, the centralized or decentralized dispatching mechanism ultimately receives a response from the server (e.g., the document requested by the client) (step **1040**). Finally, the response, typically in the form of an HTML document is forwarded to the client (step **1050**). This method of caching documents segregates the content of the Web based upon the URL of the documents. Since each URL will map to only one proxy server **405**, advantageously this approach more efficiently allocates the proxy's cache space by avoiding unnecessary redundancy.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It

will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed and desired to be secured by United States Letters Patent is:

1. In a networked computer system such as the Internet that includes a plurality of remote servers, a plurality of proxy servers and a plurality of client systems, all of which are logically interconnected so that the client systems can access informational content stored at the one or more remote servers, and wherein at least one of the client systems is comprised of an electronics unit which provides a graphical user interface by which the Internet can be accessed and browsed using a conventional television set as a display, a method of efficiently allocating cache space within the plurality of proxy servers so that requested content from one or more Web pages is distributively cached at mutually exclusive proxy servers, comprising steps for:

dividing responsibility for obtaining and caching content among a plurality of proxy servers, wherein at least two proxy servers are responsible for obtaining and caching mutually exclusive content;

receiving a request for downloading content from a particular Web page identified by a uniform resource locator ("URL");

rather than allowing any arbitrary proxy server to obtain and cache the requested content from the particular Web page identified, mapping the URL of the particular Web page to a particular one and only one of a plurality of mutually exclusive ranges of URLs that are distributed among the plurality of proxy servers; and

at a proxy server assigned the particular mutually exclusive range to which the URL of the particular Web page is mapped, searching for the requested content in a local cache, and if the requested content is found in the local cache, returning it to the client system from which the request was received, and if the requested content is not found, then obtaining the requested content from one of the remote servers and storing it in the local cache, and returning the requested content to the client system from which the request was received.

2. In a networked computer system such as the Internet that includes a plurality of remote servers, a plurality of proxy servers and a plurality of client systems, all of which are logically interconnected so that the client systems can access informational content stored at the one or more remote servers, and wherein at least one of the client systems is comprised of an electronics unit which provides a graphical user interface by which the Internet can be accessed and browsed using a conventional television set as a display, a computer program product for implementing a method of efficiently allocating cache space within the plurality of proxy servers so that requested content from one or more Web pages is distributively cached at mutually exclusive proxy servers, comprising a computer readable medium for storing executable instructions for implementing the method, and wherein the method comprises steps for:

dividing responsibility for obtaining and caching content among a plurality of proxy servers, wherein at least two proxy servers are responsible for obtaining and caching mutually exclusive content;

receiving a request for downloading content from a particular Web page identified by a uniform resource locator ("URL");

rather than allowing any arbitrary proxy server to obtain and cache the requested content from the particular Web page identified, mapping the URL of the particular Web page to a particular one and only one of a plurality of mutually exclusive ranges of URLs that are distributed among the plurality of proxy servers; and

at a proxy server assigned the particular mutually exclusive range to which the URL of the particular Web page is mapped, searching for the requested content in a local cache, and if the requested content is found in the local cache, returning it to the client system from which the request was received, and if the requested content is not found, then obtaining the requested content from one of the remote servers and storing it in the local cache, and returning the requested content to the client system from which the request was received.

3. A method as recited in claim 1 or claim 2, wherein the step for mapping identifies a cluster of two or more proxy servers collectively assigned the particular one of the plurality of mutually exclusive ranges of URLs, the method further comprising a step for storing requested content that is retrieved from one of the remote servers in the local cache of each proxy server in the cluster.

4. A method as recited in claim 3, further comprising a step for distributing multiple requests for the requested content among the two or more proxy servers of the cluster.

5. A method as recited in claim 1 or claim 2, wherein the stop for mapping the URL of the particular Web page comprises the act of applying a hash algorithm to the URL.

6. A method as recited in claim 5, wherein the hash algorithm comprises a Message Digest 5 algorithm.

7. A method as recited in claim 1 or claim 2, wherein a central dispatcher receives the request for downloading content and maps the URL of the particular Web page, the method further comprising a step for forwarding the request to the assigned proxy server.

8. A method as recited in claim 1 or claim 2, wherein an initial proxy server receives the request for downloading content, the method further comprising a step for forwarding the request to the assigned proxy server after the initial proxy server searches for the requested content in a local cache and the requested content is not found.

9. In a networked computer system such as the Internet that includes a plurality of remote servers, a plurality of proxy servers and a plurality of client systems, all of which are logically interconnected so that the client systems can access informational content stored at the one or more remote servers, and wherein at least one of the client systems is comprised of an electronics unit which provides a graphical user interface by which the Internet can be accessed and browsed using a conventional television set as a display, a method of efficiently allocating cache space within the plurality of proxy servers so that requested content from one or more Web pages is distributively cached at mutually exclusive proxy servers, comprising acts of:

assigning to each of a plurality of proxy servers a mutually exclusive range of uniform resource locators ("URL");

hashing a URL, received as part of a request from a client system for downloading content from a particular Web page;

rather than allowing any arbitrary proxy server to retrieve and cache the requested content, identifying, with at least a portion of the hashed URL, at least one proxy server assigned to a particular mutually exclusive range of URLs that corresponds to the requested content; and

at the at least one identified proxy server, examining a local cache for the requested content, and if found in

the local cache, sending the requested content to the client system, hut if not found in the local cache, retrieving the requested content from one of the remote servers, storing the requested content in the local cache, and sending the requested content to the client system.

10. In a networked computer system such as the Internet that includes a plurality of remote servers, a plurality of proxy servers and a plurality of client systems, all of which are logically interconnected so that the client systems can access informational content stored at the one or more remote servers, and wherein at least one of the client systems is comprised of an electronics unit which provides a graphical user interface by which the Internet can he accessed and browsed using a conventional television set as a display, a computer program product for implementing a method of efficiently allocating cache space within the plurality of proxy servers so that requested content from one or more Web pages is distributively cached at mutually exclusive proxy servers, comprising a computer readable medium for storing executable instructions for implementing the method, and wherein the method comprises acts of

assigning to each of a plurality of the proxy servers a mutually exclusive range of uniform resource locators ("URLs");

hashing a URL, received as part of a request from a client system for downloading content from a particular Web page;

rather than allowing any arbitrary proxy server to retrieve and cache the requested content, identifying, with at least a portion of the hashed URL, at least one proxy server assigned to a particular mutually exclusive range of URLs that corresponds to the requested content; and

at the at least one identified proxy server, examining a local cache for the requested content, and if found in the local cache, sending the requested content to the client system, hut if not found in the local cache, retrieving the requested content from one of the remote servers, storing the requested content in the local cache, and sending the requested content to the client system.

11. A method as recited in claim 9 or claim 10, wherein the act of hashing identifies a cluster of two or more proxy servers collectively assigned to the particular mutually exclusive range of URLs that corresponds to the requested content, the method further comprising an act of adding requested content that is received from one of the remote servers to the local cache of each proxy server in the cluster.

12. A method as recited in claim 11, further comprising an act of load halancing multiple requests for the requested content among the two or more proxy servers of the cluster.

13. A method as recited in claim 9 or claim 10, wherein the hashing comprises a Message Digest 5 algorithm.

14. A method as recited in claim 9 or claim 10, wherein a central dispatcher receives the request for downloading content and hashes the received URL, the method further comprising an act of sending the request to the at least one identified proxy server.

15. A method as recited in claim 9 or claim 10, wherein an initial proxy server receives the request for downloading content, the method further comprising an act of sending the request to the at least one identified proxy server after the initial proxy server searches for the requested content in a local cache and the requested content is not found.

* * * * *